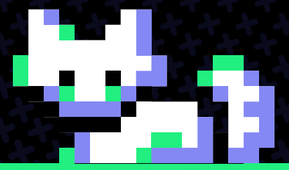
PLAYER 1

HIGHSCORE 2500

PLAYER 2

OC--

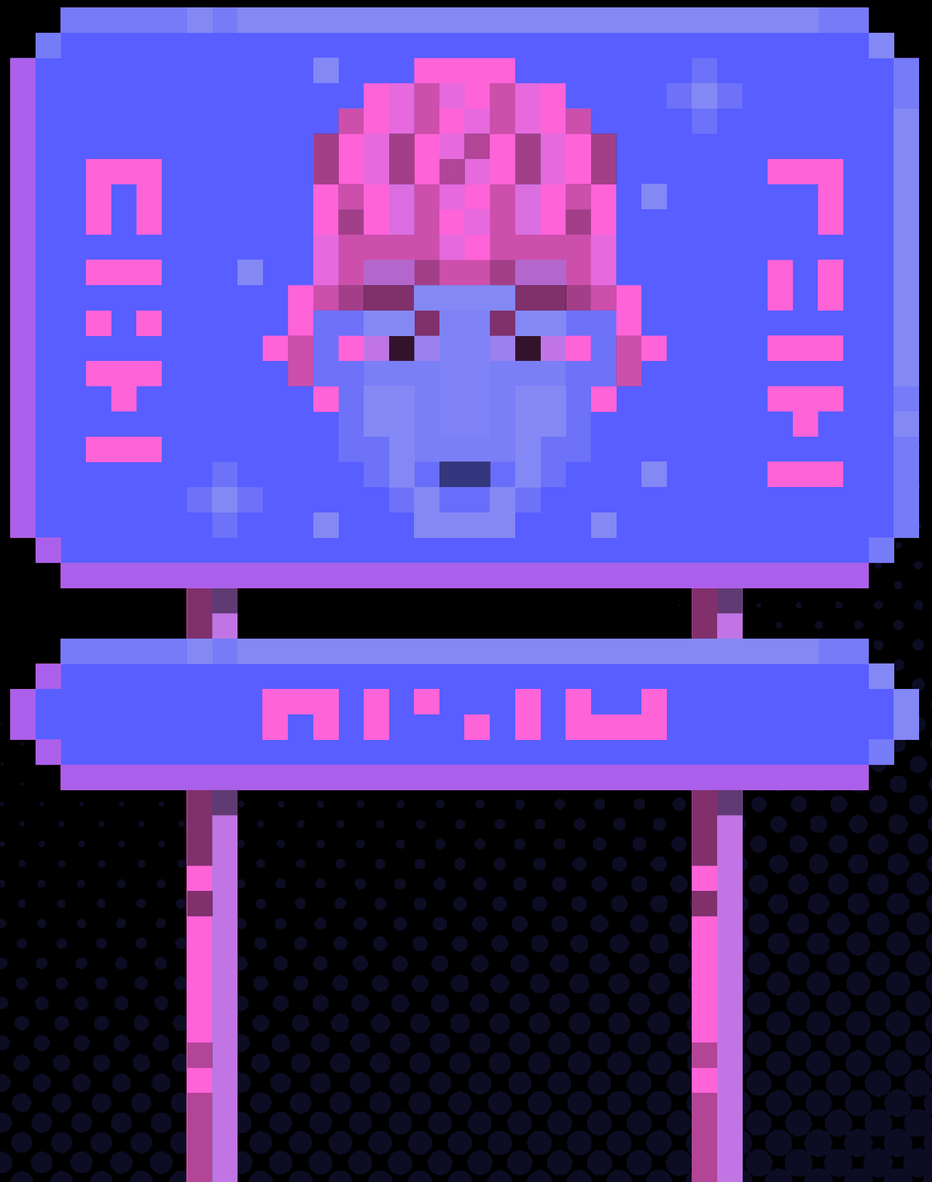
AVOID THE DISCIPLINARY COMMITTEE



MENU 01 07 12

OBJECTIVE

- THROUGHOUT THE GAME, YOU WILL FACE SEVERAL LEVELS, OR ARCS AND WILL BE GIVEN CHOICES.
- CHOOSE YOUR DECISIONS TO AVOID GETTING THE DC ACTION ON YOU.
- GOOD LUCK, HAVE FUN!!



MENU 01 07 12

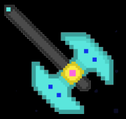
LEVELS

TOPICS COVERED

HOME ARC

CLASSROOM ARC

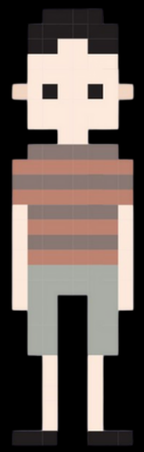
SOCIETY ARC



HALL ARC

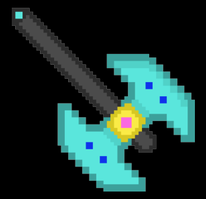
BACK TO LEVELS PAGE

HOME ARC



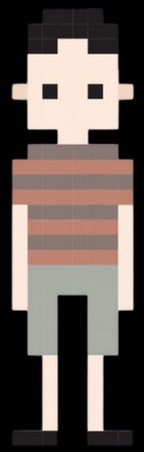
YOU :

Uhh, what time is it?



GET UP

A FEW MORE MINUTES WONT HURT...



YOU :

Shit, I’m gonna be late for class, the prof is definitely going to dereg me this time, but I don’t want to get up!

You open your eyes and look at the time

You have an email from your professor:

Dear Player1,
You have been deregistered from the course, and your dear friend “Teach HersPetS Nake” from your class has informed me of all the nice things you have said about me. Hence, you can expect a mail from the DC today.

Regards
Professor Powa Trip

GAME OVER

GO TO LAST CHECKPOINT

OH WAIT:

You don’t know how to open your eyes!
Worry not I will teach you.

Almost every graphical interface, like this one, once used(and may still be using) OpenGL to render the graphics.

BUT WHAT IS OPENGL ?

"THE GL STANDS FOR GOOD LUCK BECAUSE YOU'RE GONNA NEED IT"

1

OpenGL is mainly considered to be a graphics API widely used to render the graphics we see on the screen in many popular software.

2

However, OpenGL by itself is not an API but merely a specification developed and maintained by the Khronos Group.

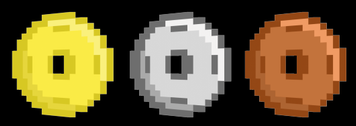
3

The OpenGL specification specifies exactly what the result/output of each function should be and how it should perform.

WHO MAKES IT?

The Spanish Inquisition, unexpected, right?

- Well, frankly, anyone who needs it.
- GPU manufacturers often tweak and create new libraries as extensions, in order for OpenGL functions to be compatible with the code
- This also means that whenever OpenGL is showing weird behavior that it shouldn't, this is most likely the fault of the graphics cards manufacturers.



YOU USE OPENGL TO SET THE VIEWPORT AND NOW FINALLY YOU CAN SEE

In OpenGL, a viewport defines the rectangular area of the window where the final rendered image is displayed. It essentially maps normalized device coordinates (NDC), which range from [−1,1][-1,1][−1,1] in both x and y axes, to window coordinates (pixel coordinates).

```
void glViewport(GLint x, GLint y, GLsizei width, GLsizei height);
```


RUN TO CLASS
NEIGHBOURS ROOM
YOU :
Finally I’m up, what do I do now?

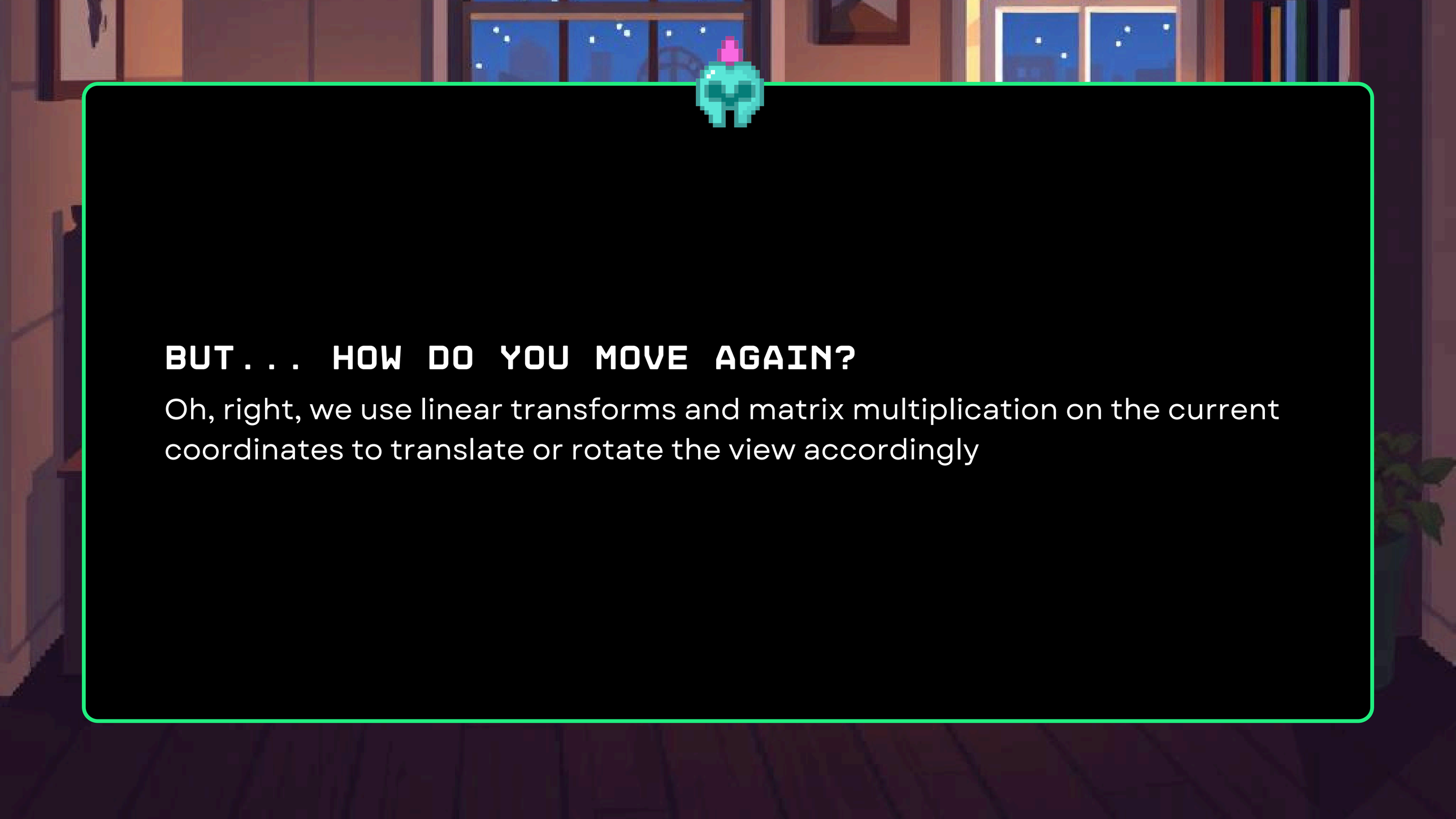
You keep knocking at the door. After 20 minutes, his roomie opens the door, says that your batchie Teach HersPetS Nake already left. You call him for proxy, he does not pick up, but you get an email:

Dear Player1,
You have been deregistered from the course, and an anonymous student from your class has informed me of all the nice things you have said about me. Hence, you can expect a mail from the DC today.

Regards
Professor Powa Trip

GAME OVER

GO TO LAST CHECKPOINT


BUT... HOW DO YOU MOVE AGAIN?
Oh, right, we use linear transforms and matrix multiplication on the current coordinates to translate or rotate the view accordingly

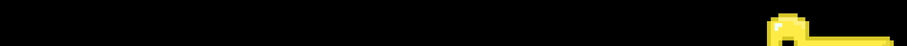
TRANSFORMS: TRANSLATION

This transform can be applied by algebraically adding to the matrices. This is represented as a matrix product as:

$$\begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + T_x \\ y + T_y \\ z + T_z \\ 1 \end{pmatrix}$$

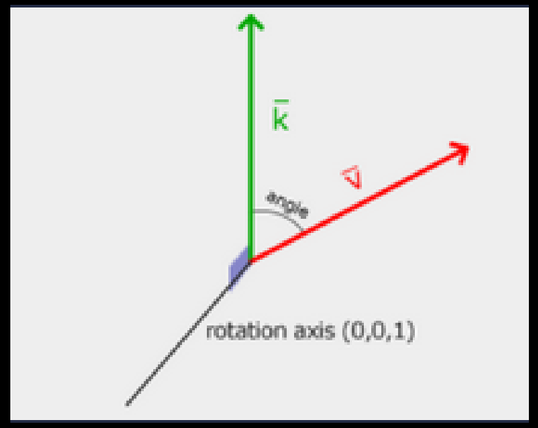
Hence, the transform matrix is obtained as :

$$\begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



TRANSFORMS: ROTATION

Using this axis system as reference:



We obtain, as a matrix product:

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta \cdot x - \sin\theta \cdot y \\ \sin\theta \cdot x + \cos\theta \cdot y \\ z \\ 1 \end{pmatrix}$$

BUT WHY MATRIX OPERATIONS?

Are we all really a part of a simulation?

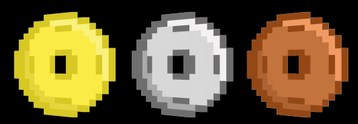
- Well, that is because matrix multiplications and operations allow us to perform multiple operations at the same time using Parallel computing.
- Since GPUs excel at parallel computing, they are well suited for these tasks.
- For Example: This is the time taken by GPU vs CPU(single core) for a random 10000 * 10000 matrix multiplication

Time taken by GPU:

```
Elapsed time is 0.003345 seconds.
```

Time taken by CPU:

```
Elapsed time is 14.283674 seconds.
```



YOU MULTIPLY YOUR CURRENT COORDINATES WITH THE REQUIRED COORDINATES AND START RUNNING TOWARDS THE SCHOOL

In OpenGL, we use the glm library to create, manipulate and operate with vectors and matrices.

```
glm::vec4 vec(1.0f, 0.0f, 0.0f, 1.0f);
glm::mat4 trans = glm::mat4(1.0f);
trans = glm::translate(trans, glm::vec3(1.0f, 1.0f, 0.0f));
vec = trans * vec;
trans = glm::rotate(trans, glm::radians(90.0f), glm::vec3(0.0, 0.0, 1.0));
trans = glm::scale(trans, glm::vec3(0.5, 0.5, 0.5));
```

BACK TO LEVELS MENU

01 06 01

CONGRATULATIONS! YOU HAVE CLEARED THE HOME ARC!!

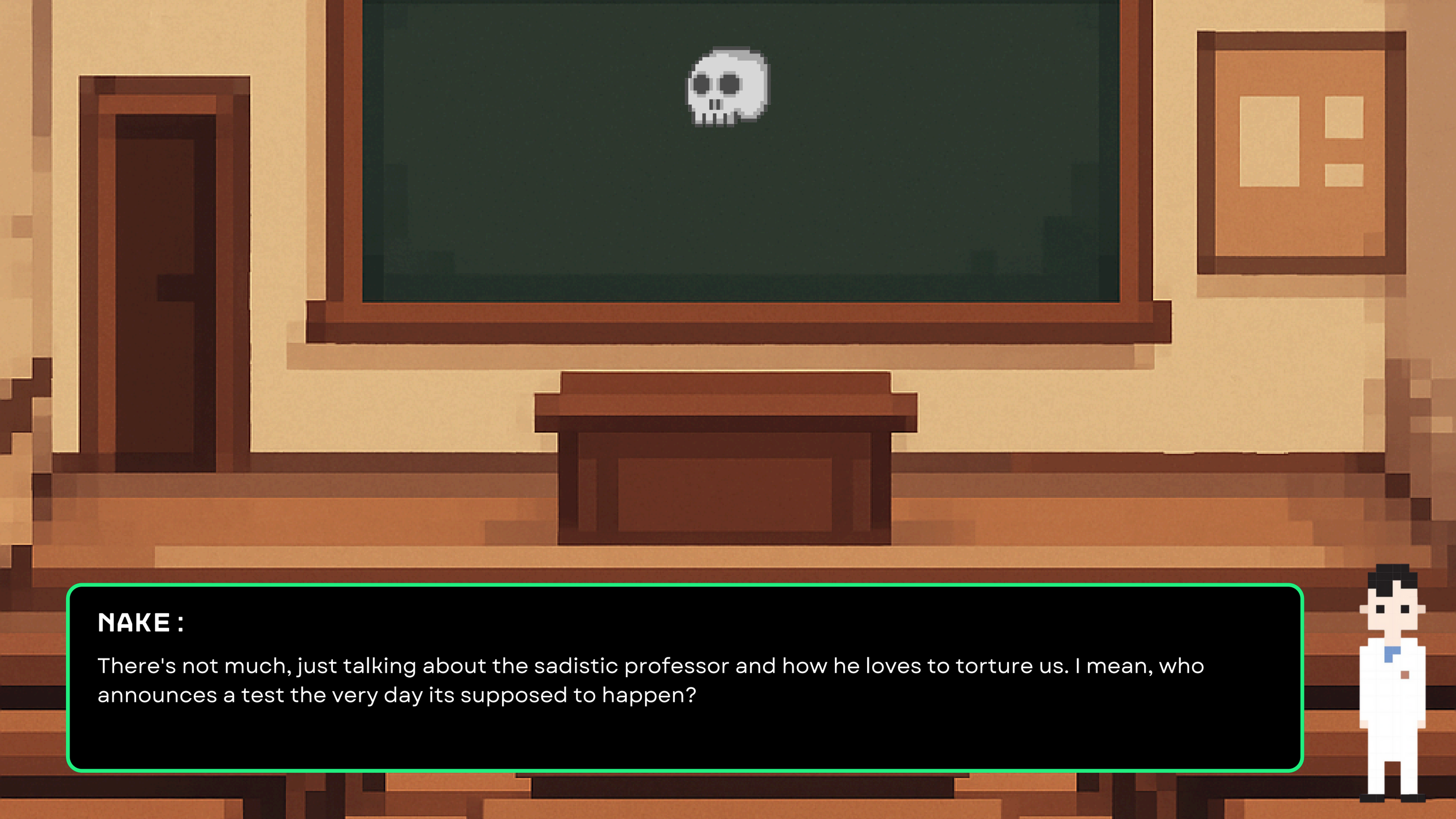
BACK TO LEVELS PAGE

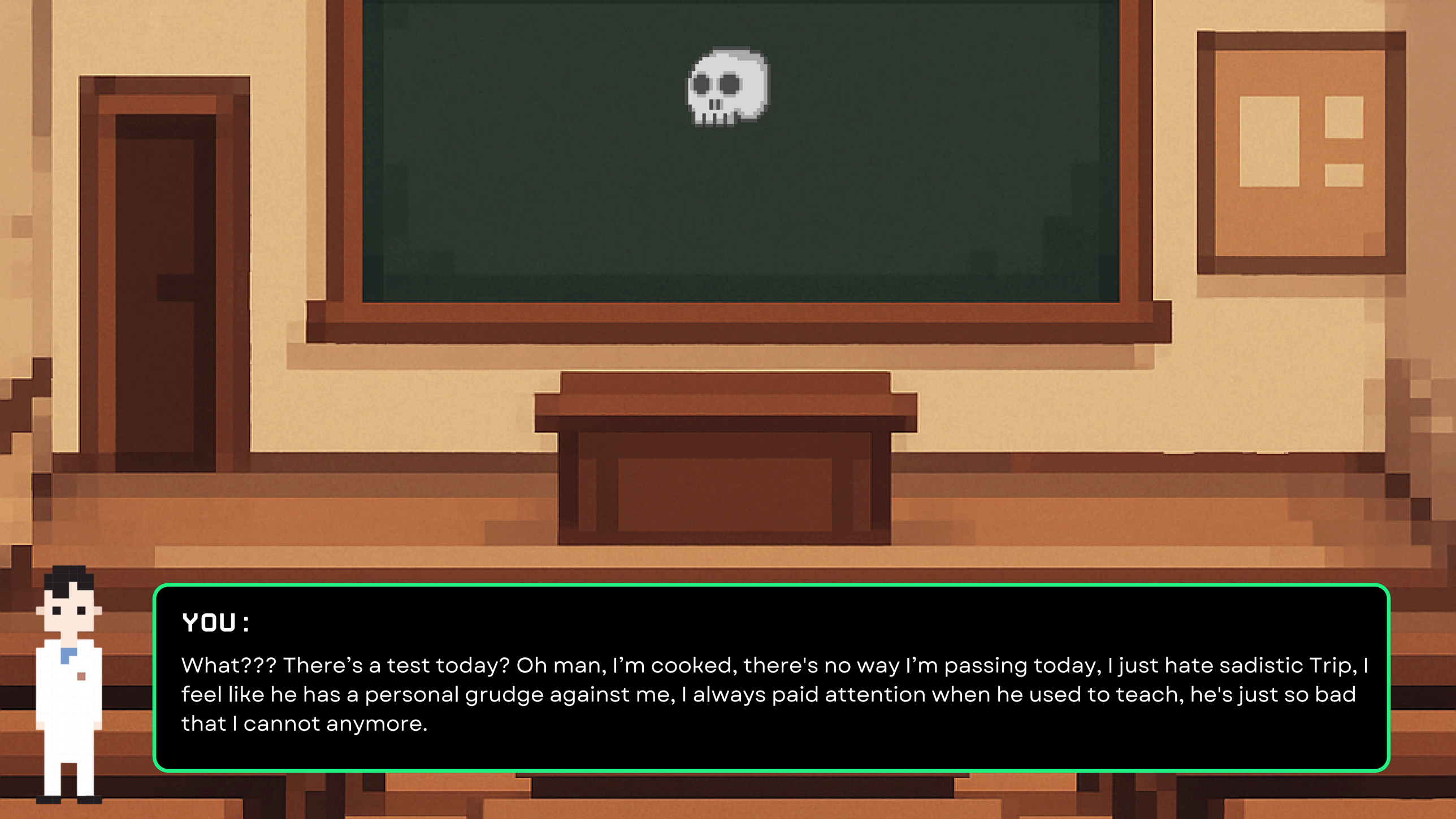
CLASSROOM ARC

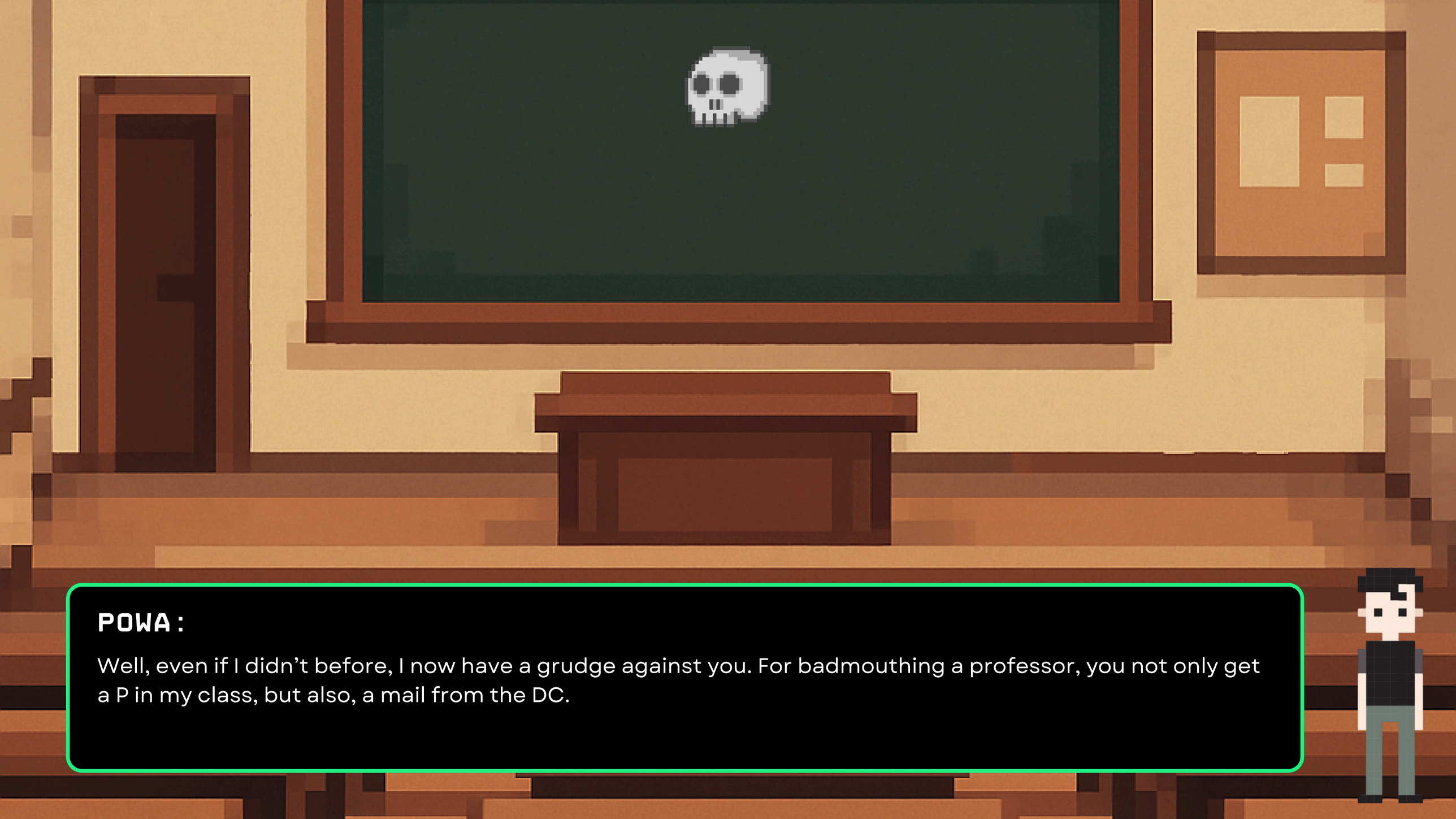

TALK TO FRIENDS
REVIEW CLASS MATERIAL
YOU :
Barely got here before time, YES! The prof is not here yet, I still have time to kill

YOU START TALKING TO NAKE

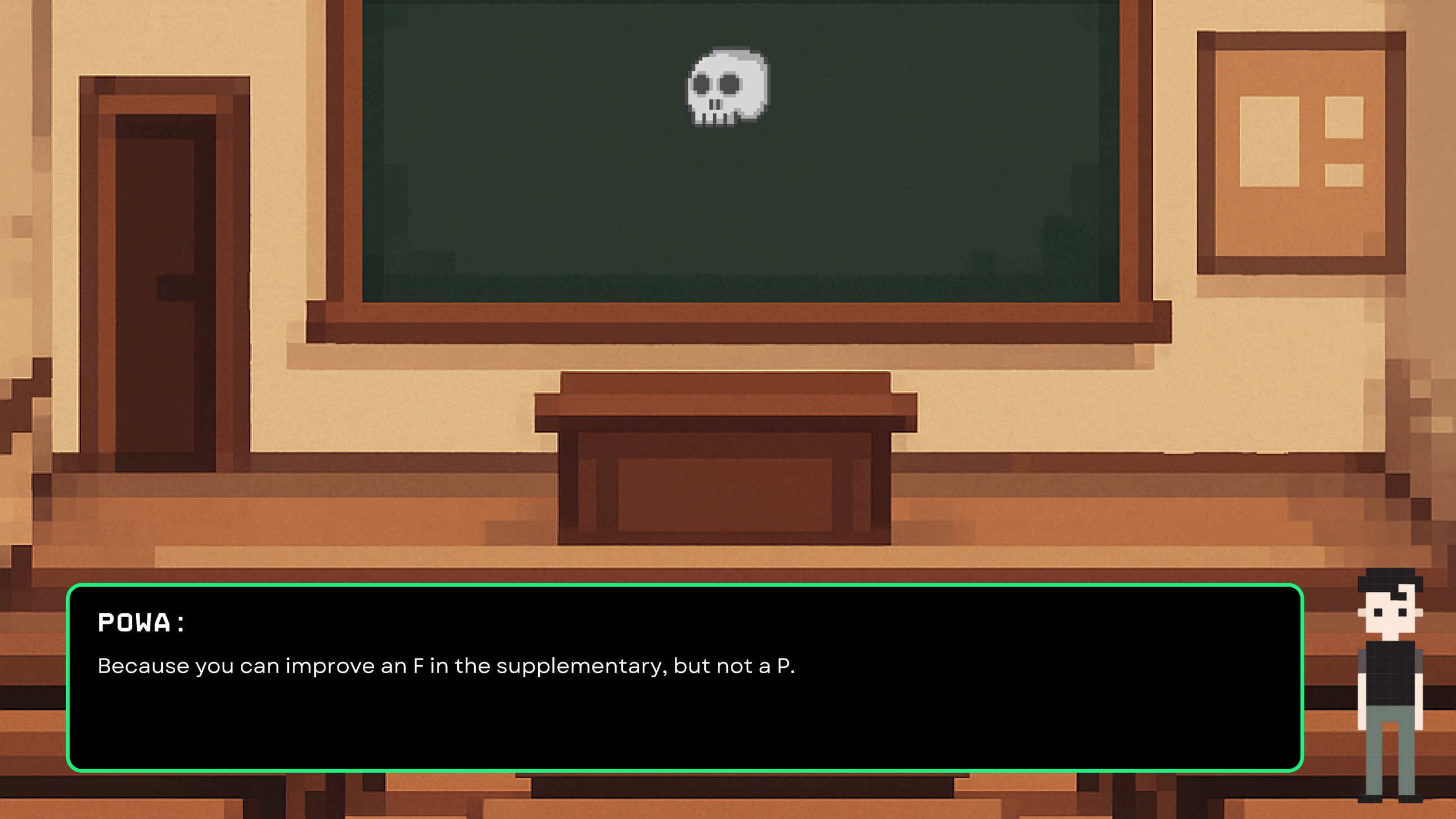

YOU :
Hey man, what’s up?


NAKE :
There's not much, just talking about the sadistic professor and how he loves to torture us. I mean, who announces a test the very day its supposed to happen?


YOU :
What??? There’s a test today? Oh man, I’m cooked, there's no way I’m passing today, I just hate sadistic Trip, I feel like he has a personal grudge against me, I always paid attention when he used to teach, he's just so bad that I cannot anymore.


POWA :
Well, even if I didn’t before, I now have a grudge against you. For badmouthing a professor, you not only get a P in my class, but also, a mail from the DC.


YOU :
What? Why a P, just give me an F


POWA :
Because you can improve an F in the supplementary, but not a P.

GAME OVER

GO TO LAST CHECKPOINT

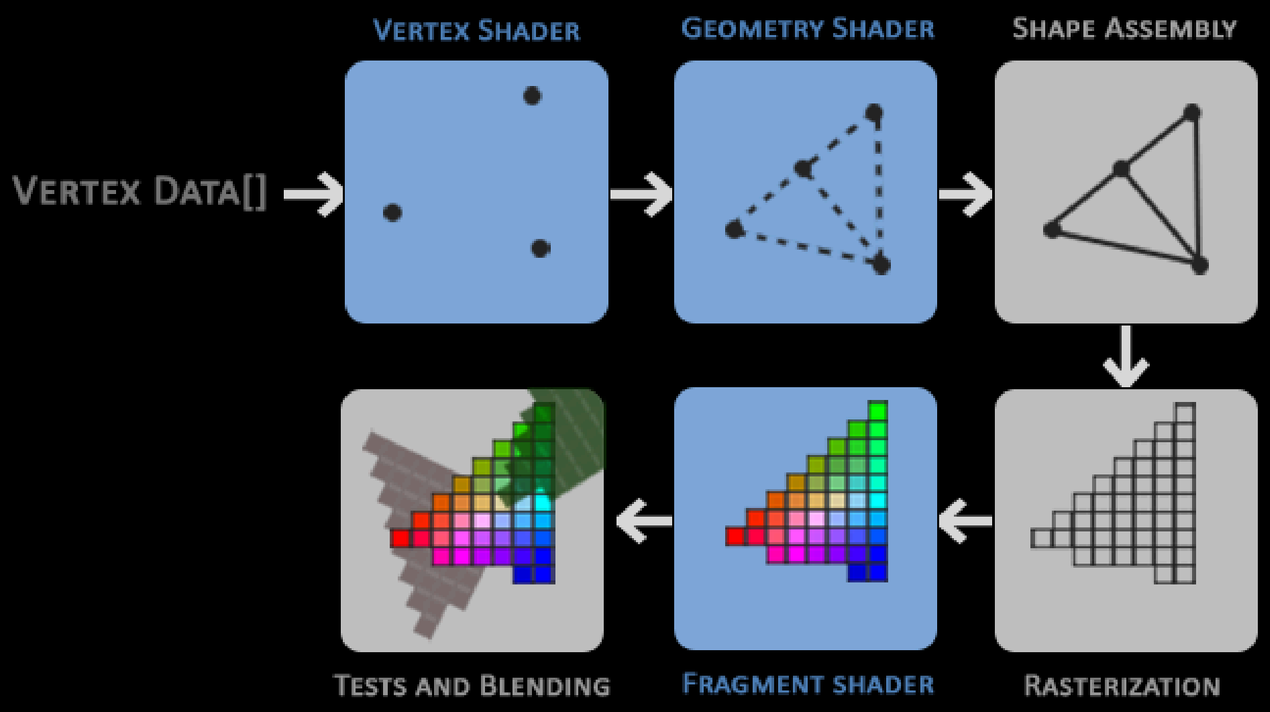
YOU GO TO READ A BOOK, BUT YOU FORGOT!! YOU HAVE NO BOOK!! WHAT DO YOU DO?

Simple, if you forgot the book, just make one!! Using OpenGL
Any render in OpenGL follows the same basic steps, known as:

OPENGL RENDER PIPELINE

RENDER PIPELINE

"I couldn't think of a punchline here, so I punched a pipe"



RENDER PIPELINE

VERTEX SHADER

The vertex shader is defined, written and linked to the program

GEOMETRY SHADER

This is an optional step that is used to apply some geometric transforms to the output of the vertex shaders

SHAPE ASSEMBLY

In this step, the program takes as input all the vertices (or vertex if GL_POINTS is chosen) from the vertex (or geometry) shader that form one or more primitives and assembles all the point(s) in the primitive shape

RENDER PIPELINE

RASTERIZATION

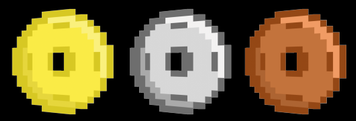
The output of the primitive assembly stage is then passed on to the rasterization stage where it maps the resulting primitive(s) to the corresponding pixels on the final screen, resulting in fragments for the fragment shader to use.

FRAGMENT SHADER

This step calculates the color values of each pixel and maps the, to the associated pixel. Most OpenGL effects are applied in this step

TESTS AND BLENDING

This stage checks the fragment's corresponding depth (and alpha) value and uses those to check if the resulting fragment is in front or behind other objects and should be discarded(or blended) accordingly.

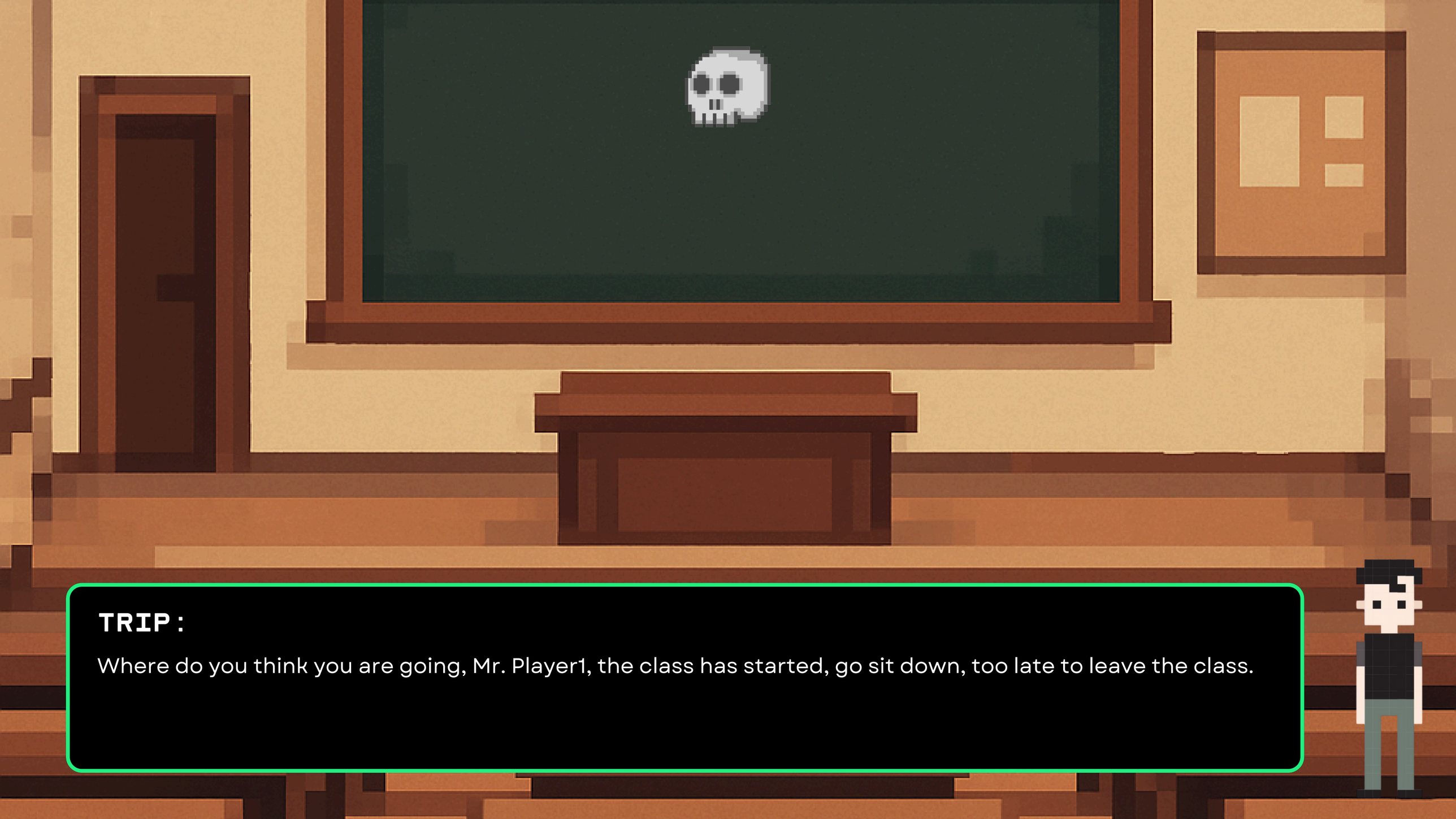


YOU LEARNT ABOUT THE RENDER PIPELINE, NOW ALL THAT'S LEFT IS TO DESIGN AND GET THE BOOK

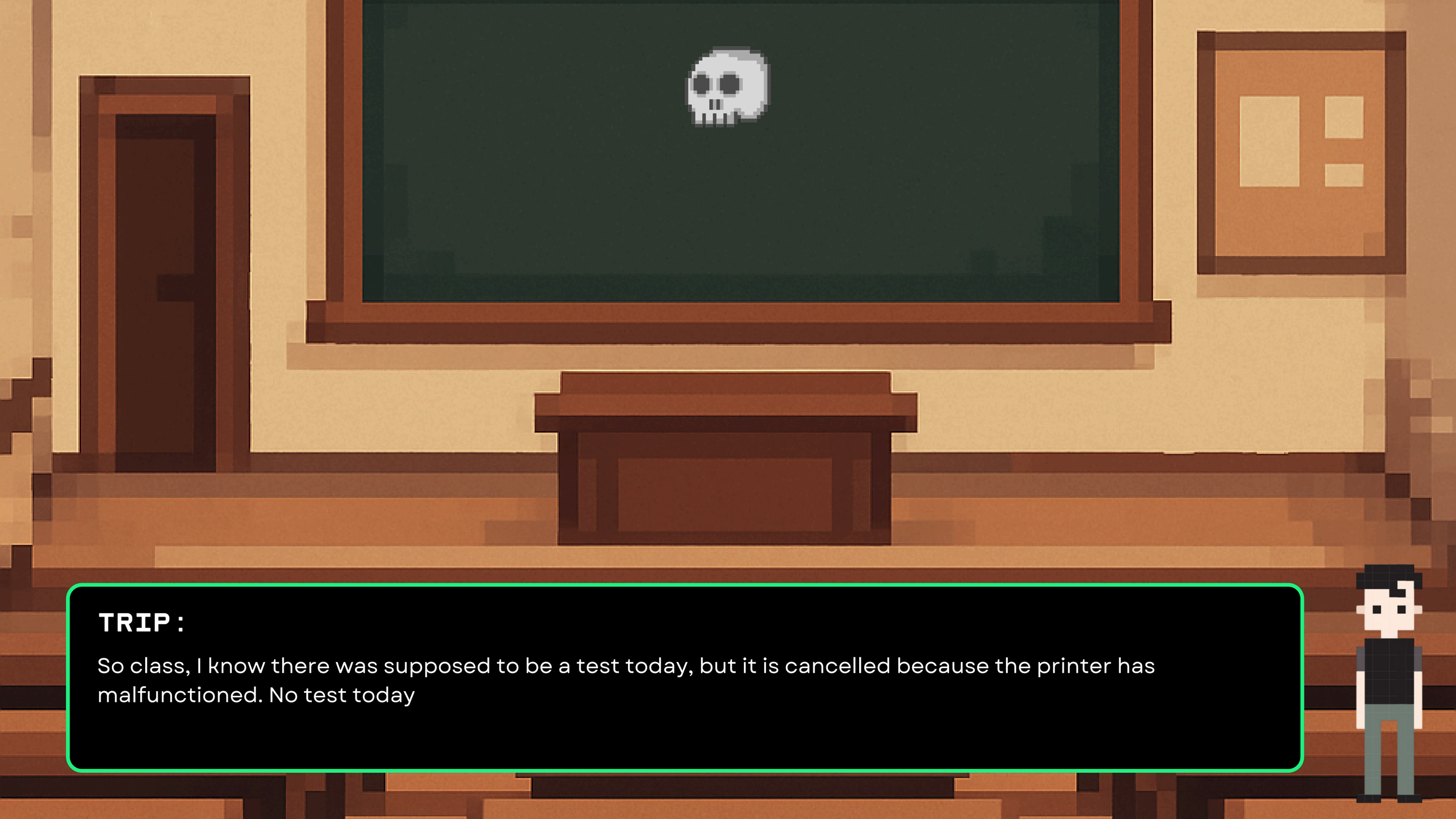
Note that some steps of the render pipeline are optional and may or may not be used.


BORROW BOOK
GO TO THE 3D PRINTER
YOU :
Now I know how to make the book... WAIT, HOW DO I DO ALL THAT, what should I do?

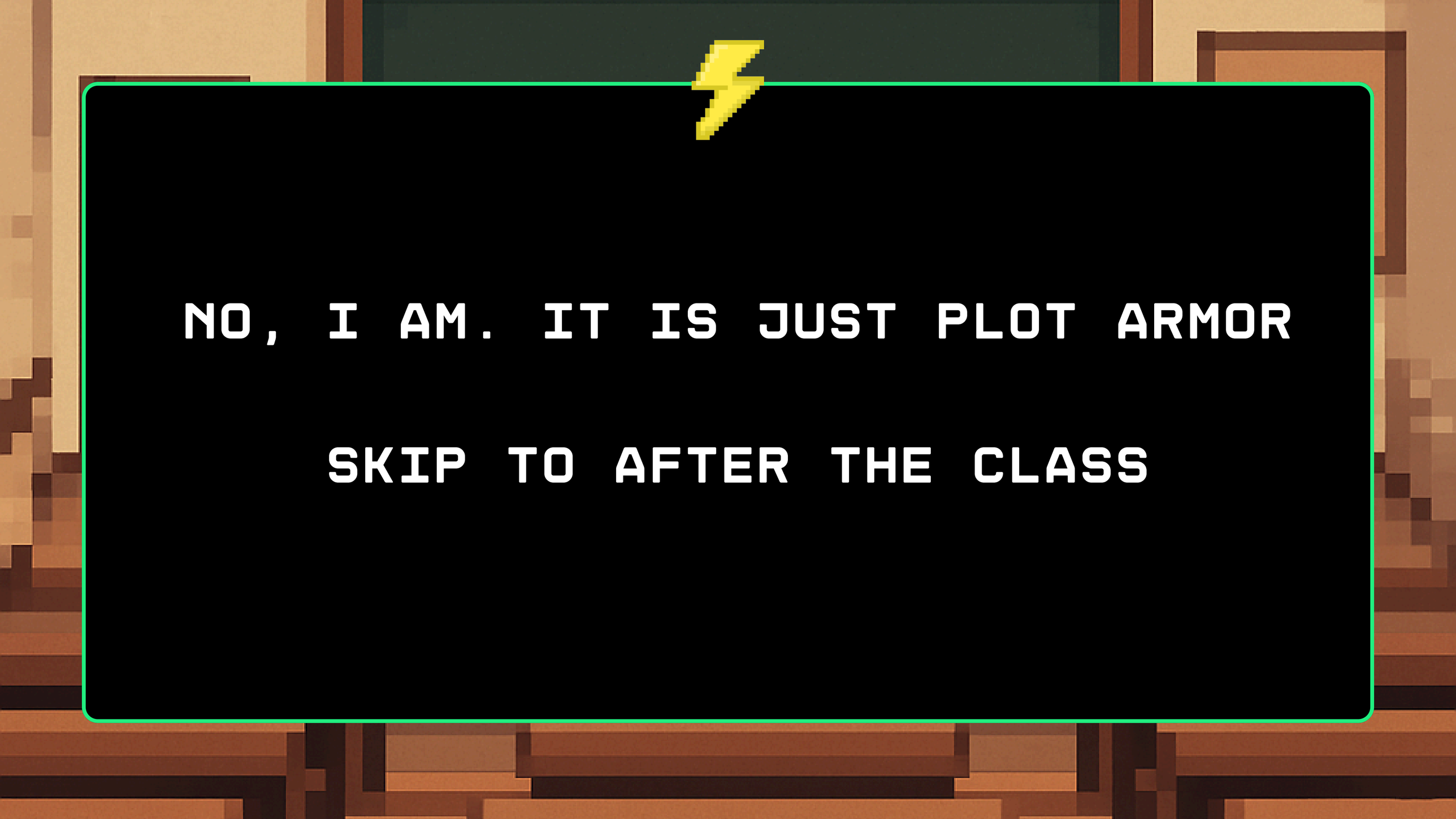
YOU START GOING TO
THE 3D PRINTER


TRIP :
Where do you think you are going, Mr. Player1, the class has started, go sit down, too late to leave the class.


YOU :
No sir, I was just dropped my eraser... (Acts to pick something up and goes back)


TRIP :
So class, I know there was supposed to be a test today, but it is cancelled because the printer has malfunctioned. No test today


YOU :
I must have been a very good person, god is protecting me.

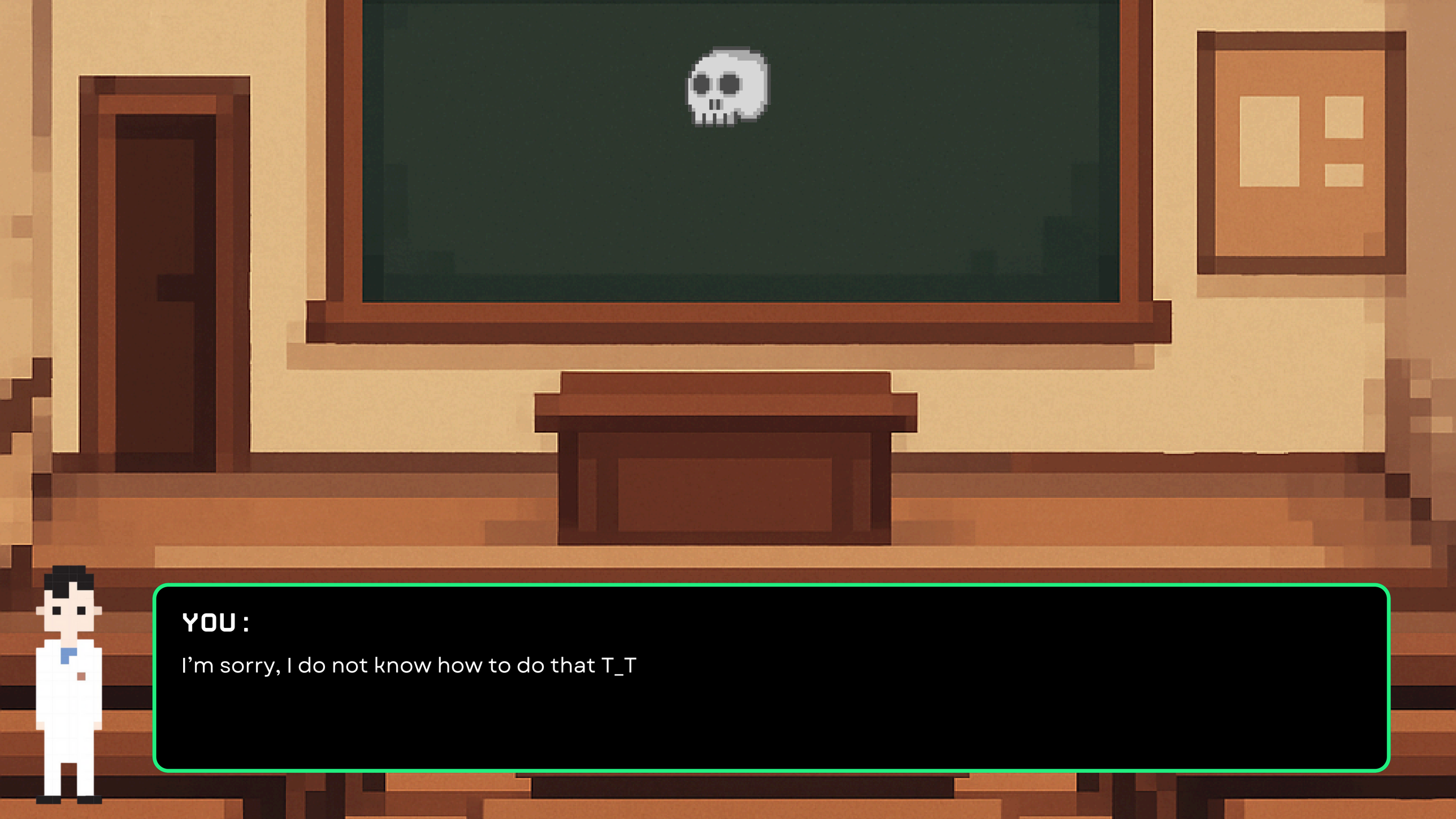

NO, I AM. IT IS JUST PLOT ARMOR
SKIP TO AFTER THE CLASS

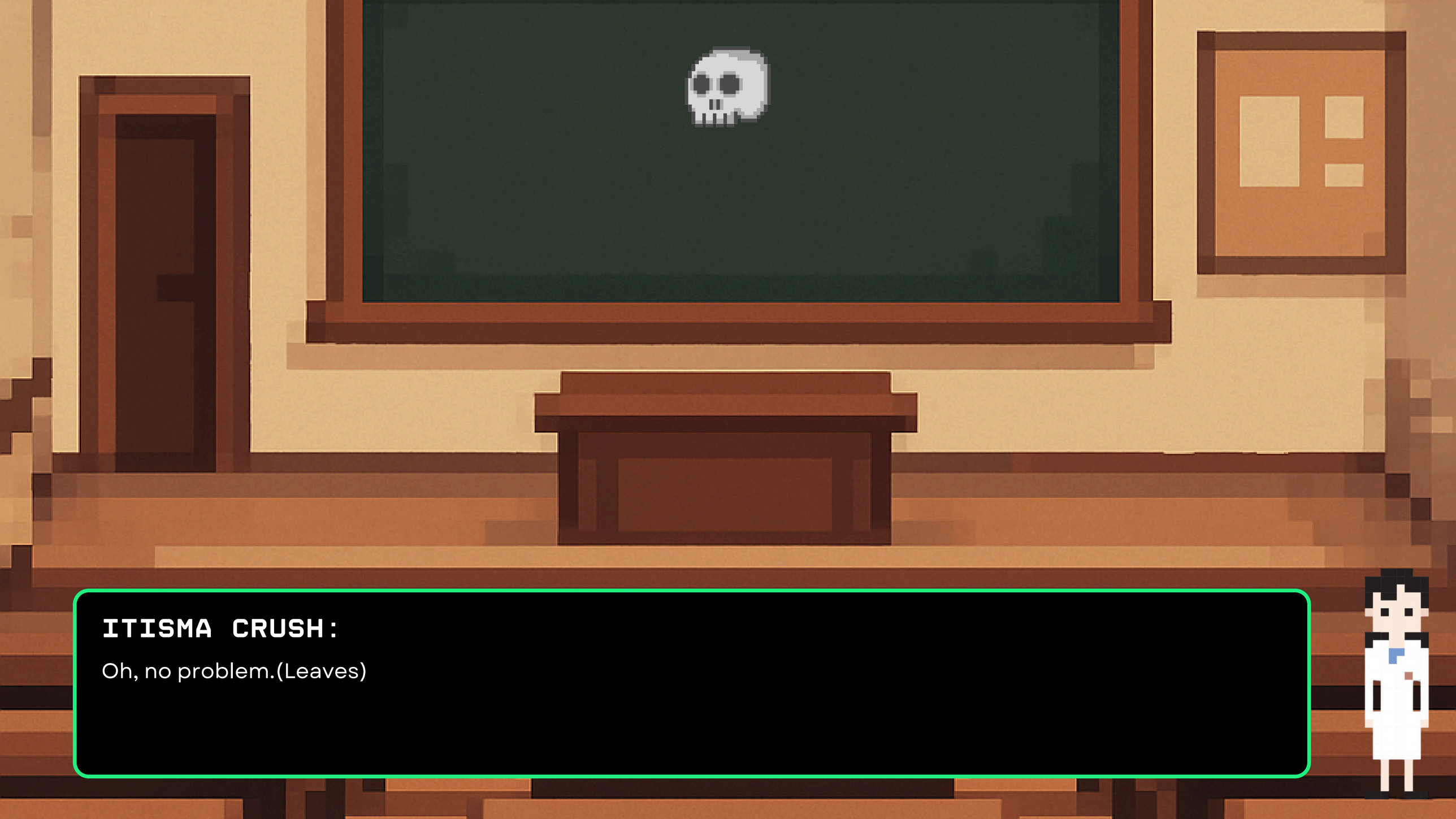

ITISMA CRUSH:
Hey Player1, what are you up to?


YOU :
Heyyyyyyy, n-nothing much mostly free whats up?


ITISMA CRUSH:
Listen, I am doing a 3D modelling project, today, but I have no idea how to do it in OpenGL, could you pleeeease help me with it?


YES! ANYTHING FOR YOU
SORRY I DON'T KNOW HOW TO DO THAT
YOU:
...


YOU :
I’m sorry, I do not know how to do that T_T


ITISMA CRUSH:
Oh, no problem.(Leaves)


YOU :
Is this how my love story ends?

YES

YOU NEVER TALKED AGAIN

GAME OVER

GO TO LAST CHECKPOINT

YES

BUT I MAKE THE RULES


YOU :
YEAH! Sure, anything for yo- I mean its cool, its cool.

AH TO BE YOUNG

BUT HOW ARE YOU GONNA TEACH HER WHEN YOU DON'T EVEN KNOW WHAT SHADERS ARE?

I GUESS I'LL HAVE TO TEACH YOU

ABOUT THE S(H)ADISTIC PART OF OPENGL

S(H)ADISTIC OPENGL

“I know what I wrote”

- Shaders are little programs that rest on the GPU.
- These programs are run for each specific section of the graphics pipeline. In a basic sense, shaders are nothing more than programs transforming inputs to outputs.
- Shaders are also very isolated programs in that they're not allowed to communicate with each other
- The only communication they have is via their inputs and outputs.

ENTONCES, ¿QUÉ LENGUAJE ENTIENDEN LOS SHADERS?

“So what language do shaders understand?”

- Shaders are written in a C-like syntax, and are generally stored in the form of .txt files, which are then fed to the program by using standard file input and output.
- This syntax is called “GLSL”
- GLSL is tailored for use with graphics and contains useful features specifically targeted at vector and matrix manipulation
- It has libraries like “glm” that have pre-defined implementation of commonly used mathematical operations and objects.

GLSL STRUCTURE

"Get Shady and face the Aftermath"

VERSION AND INPUT VARIABLE DECLARATION

In this part, we define the version number to be used for the shader and the input variables that need to be passed to the shader.

DEFINITION OF OUTPUT VARIABLES

Here, we tell OpenGL what out shader will output, as that will be passed as input to the next shader in the pipeline

DECLARATION OF OTHER VARIABLES

In this part, we define **Uniforms,** which are used to send data from the CPU to the GPU. (The input variables send data across shaders). We can also define Helper Variables.

MAIN FUNCTION AND OUTPUT VARIABLES

Here, we define what operations need to be performed before assigning values to the output variables inside a main() function.

ACCESSING SHADERS

Go Fetch! (the Shaders)

CREATING A WINDOW

We create a window using GLFW. This window is where all the graphics output shows up.

MAKING THE SHADER

We read the shader file and extract the text. We store it as a C string as OpenGL supports that format. We then store all the shaders in a sequence as each shader is characterised by its address, an unsigned int

INTERMEDIATES

This part varies from program to program. The most common implementation is usage of a Triangle Mesh which uses the following snippet to draw the triangle

```
glBindVertexArray(VAO);
glDrawArrays(GL_TRIANGLES, 0, vertex_count);
```

CLEAN UP

In this part, we delete the shader program we created and free the memory

FOUR SHADE(R)S OF OPENGL

NSFW Warning(excessive crying)

- Shaders are of (mainly) four types:
  a. Vertex Shaders
  b. Fragment Shaders
  c. Geometry Shaders
  d. Compute Shaders

- Additionally, there are two more types of shaders:
  - Tessellation Control Shaders
  - Tessellation Evaluation Shaders

VERTEX SHADERS

"You need to be sharp to implement these"

- The Vertex Shader is the programmable Shader stage in the rendering pipeline that handles the processing of individual vertices.
- Vertex shaders are fed Vertex Attribute data, as specified from a vertex array object by a drawing command.
- A vertex shader receives a single vertex from the vertex stream and generates a single vertex to the output vertex stream.
- There must be a 1:1 mapping from input vertices to output vertices.
- User-defined input values to vertex shaders are sometimes called "vertex attributes". Their values are provided by issuing a drawing command while an appropriate vertex array object is bound.
- Output variables from the vertex shader are passed to the next section of the pipeline.

FRAGMENT SHADERS

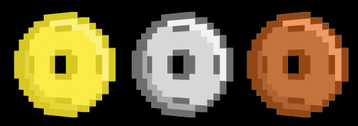
- A fragment shader is a stage in the OpenGL pipeline that processes fragments (potential pixels) generated during rasterization. It takes interpolated vertex data as input and outputs:
  - Depth value
  - Color values for the framebuffer
  - Stencil value (unmodified)
- The shader operates per fragment, allowing for operations like texturing and lighting, and is crucial for rendering detailed graphics.

GEOMETRY SHADERS

- A Geometry Shader (GS) is a Shader program written in GLSL that governs the processing of Primitives.
- The main reasons to use a GS are:
  - Layered rendering: taking one primitive and rendering it to multiple images without changing bound render targets.
  - Transform Feedback: This is often employed for computational tasks on the GPU (obviously pre-Compute Shader).

COMPUTE SHADERS

- A Compute Shader is a special type of shader in OpenGL designed for general-purpose parallel computing on the GPU.
- Unlike other shaders (vertex, fragment, geometry), compute shaders do not operate within the graphics pipeline—instead, they are executed as standalone programs for tasks like physics simulations, image processing, and AI.



NOW YOU KNOW ABOUT OPENGL SHADERS. YOU CAN PROCEED WITH YOUR TEEN ROMANCE

But since no romance is allowed in the classroom, let's move to your society

BACK TO LEVELS MENU

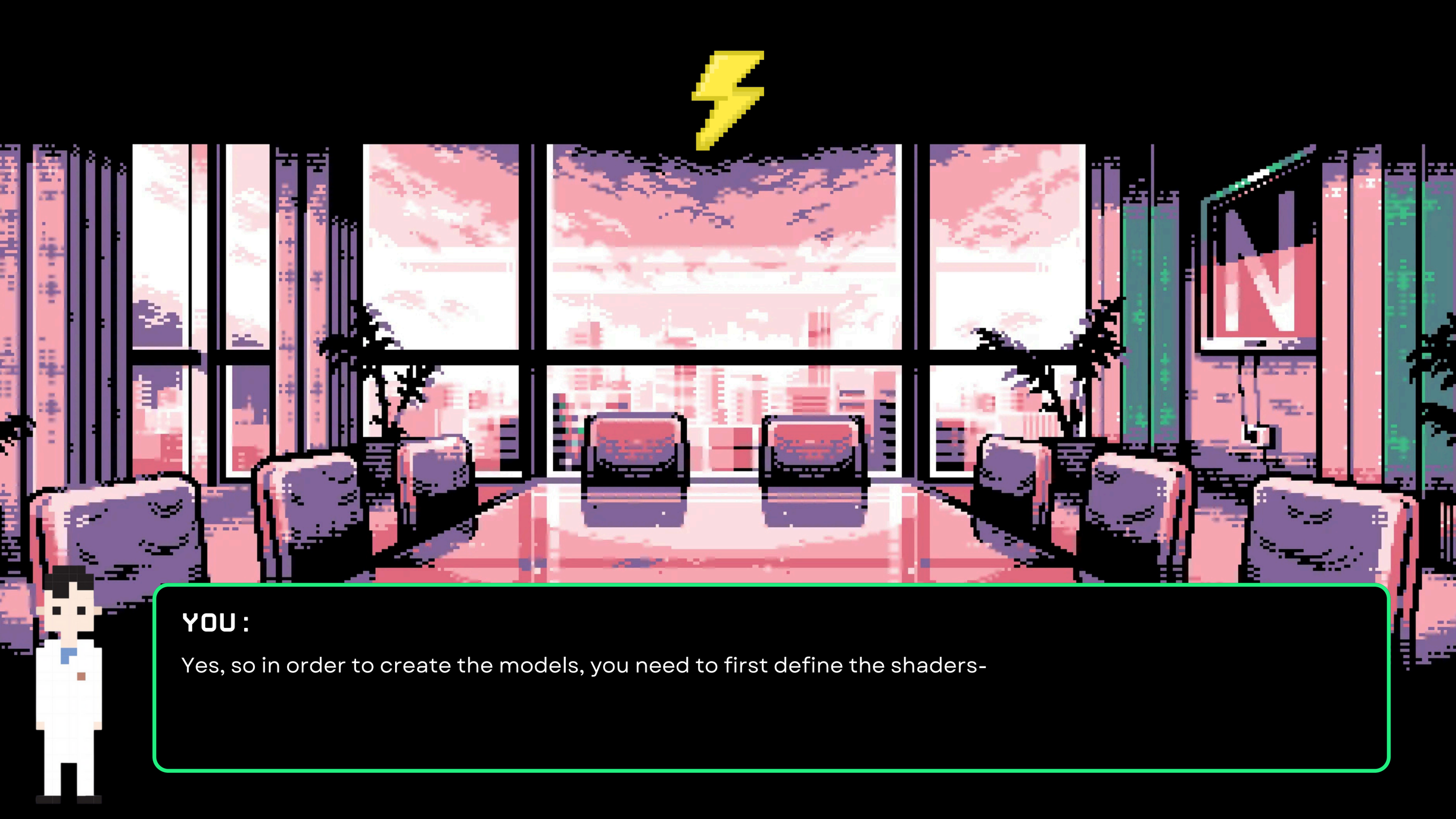
02 12 02

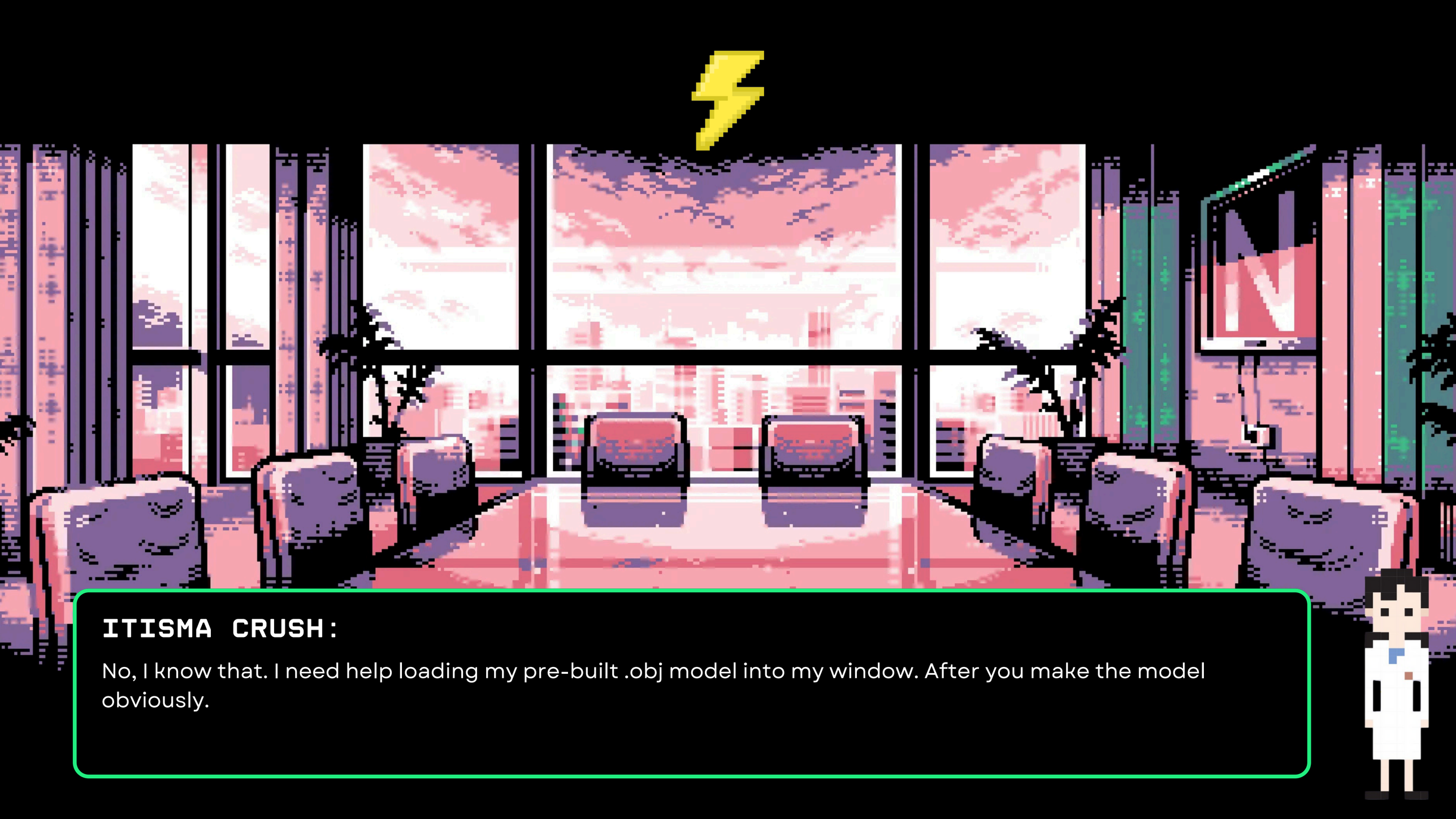
CONGRATULATIONS! YOU HAVE CLEARED THE CLASSROOM ARC!!

BACK TO LEVELS PAGE

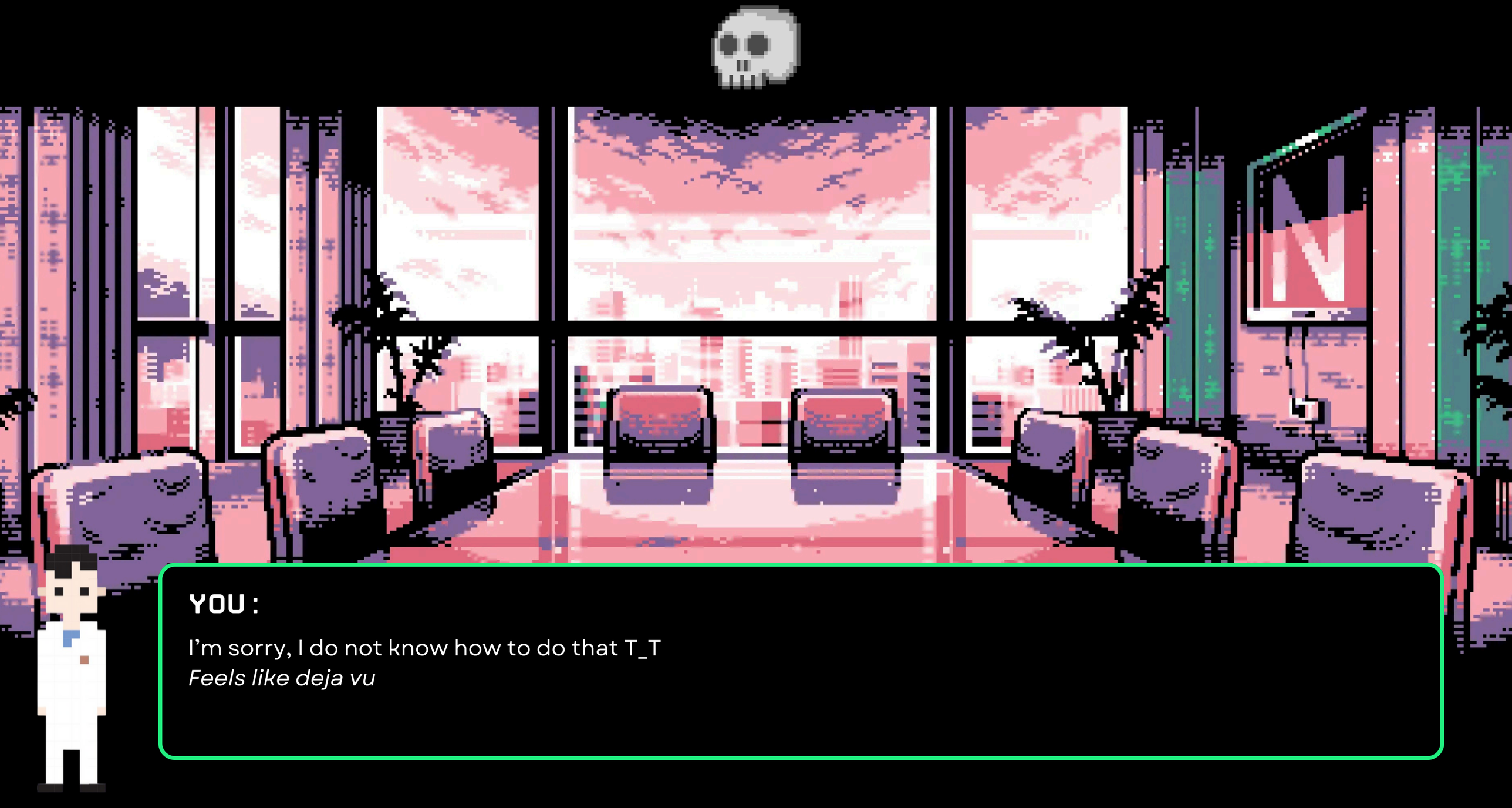
SOCIETY ARC

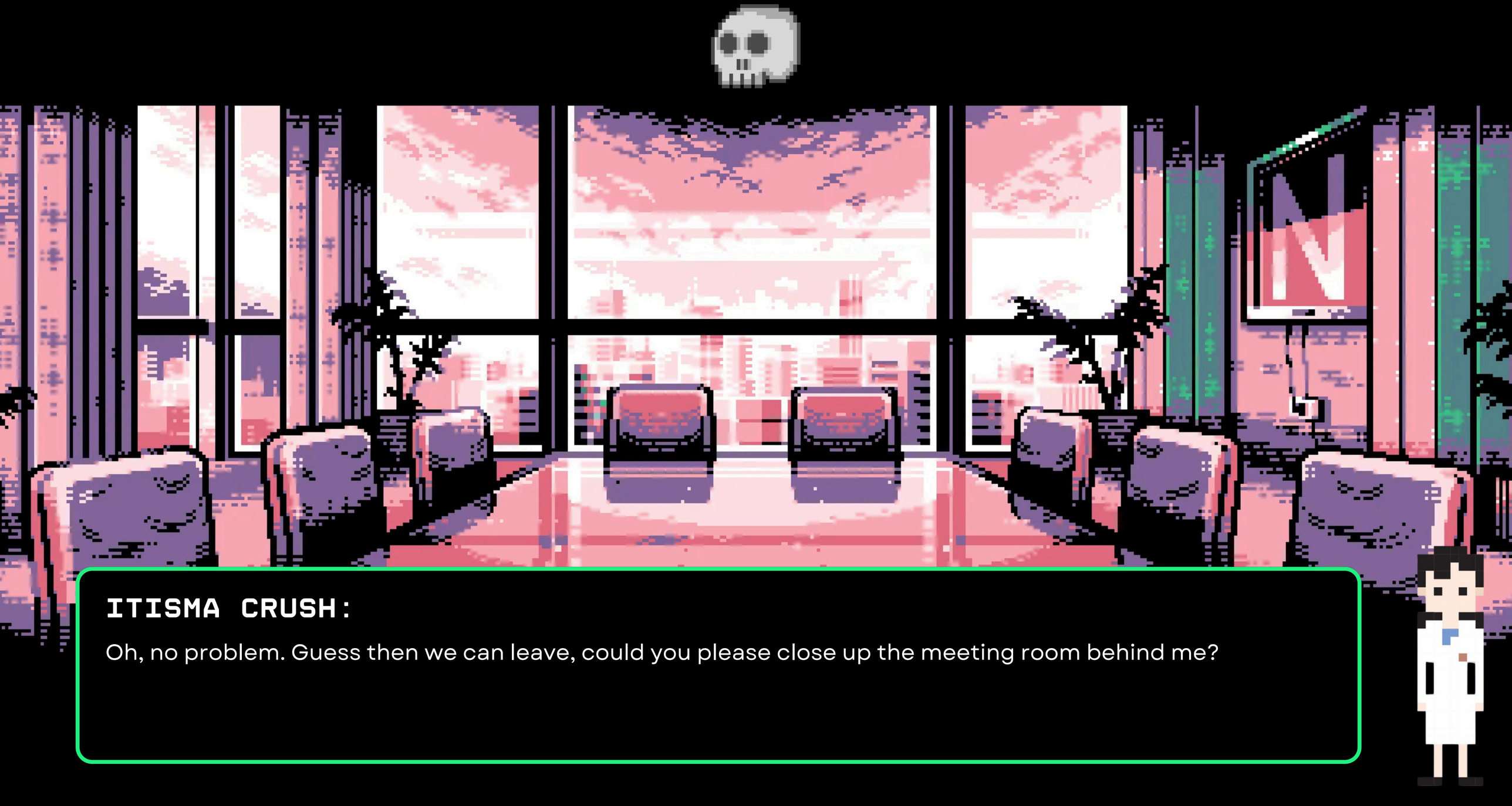

ITISMA CRUSH:
Hey Player1, are you ready?

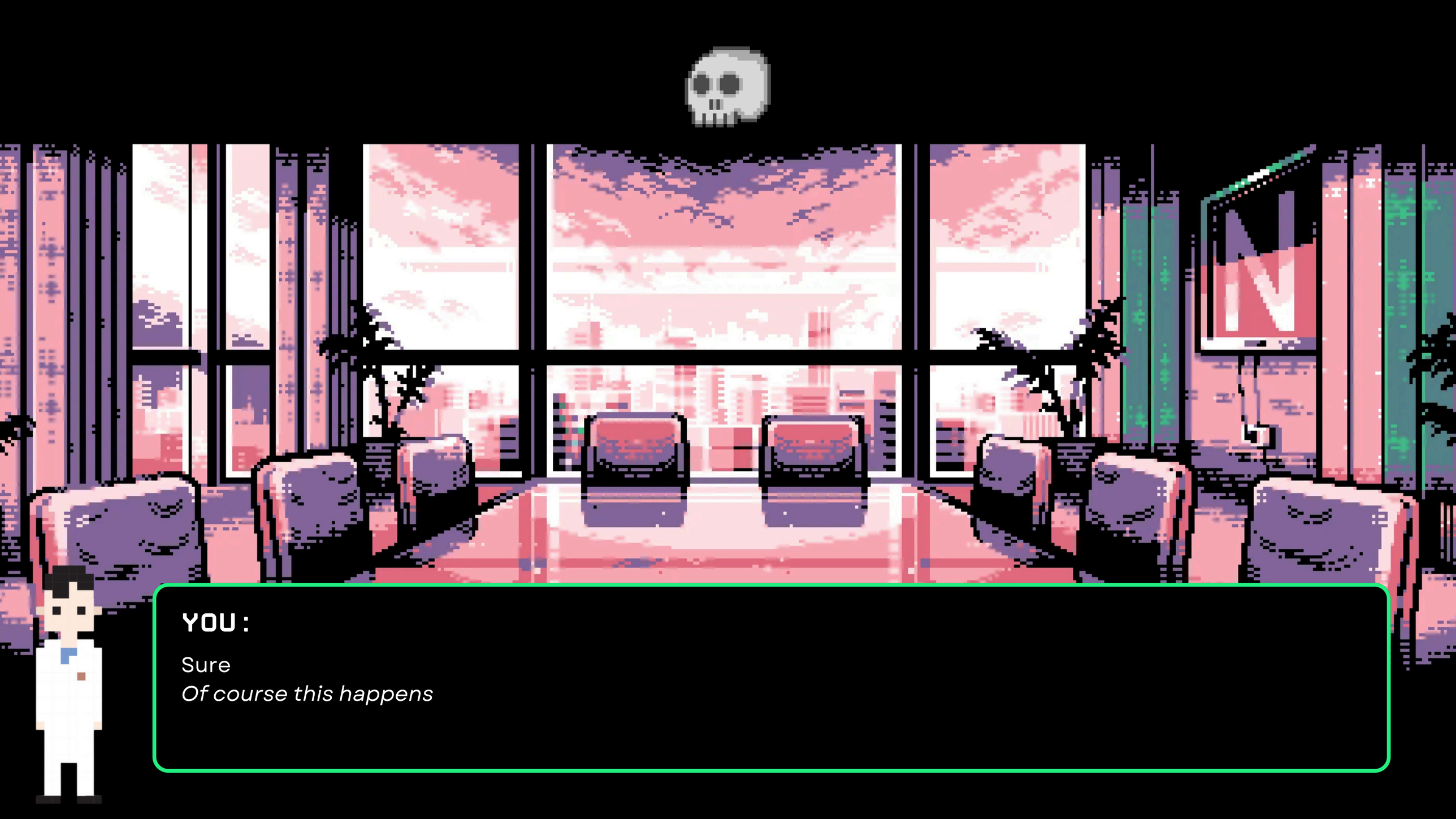

YOU :
Yes, so in order to create the models, you need to first define the shaders-


ITISMA CRUSH:
No, I know that. I need help loading my pre-built .obj model into my window. After you make the model obviously.

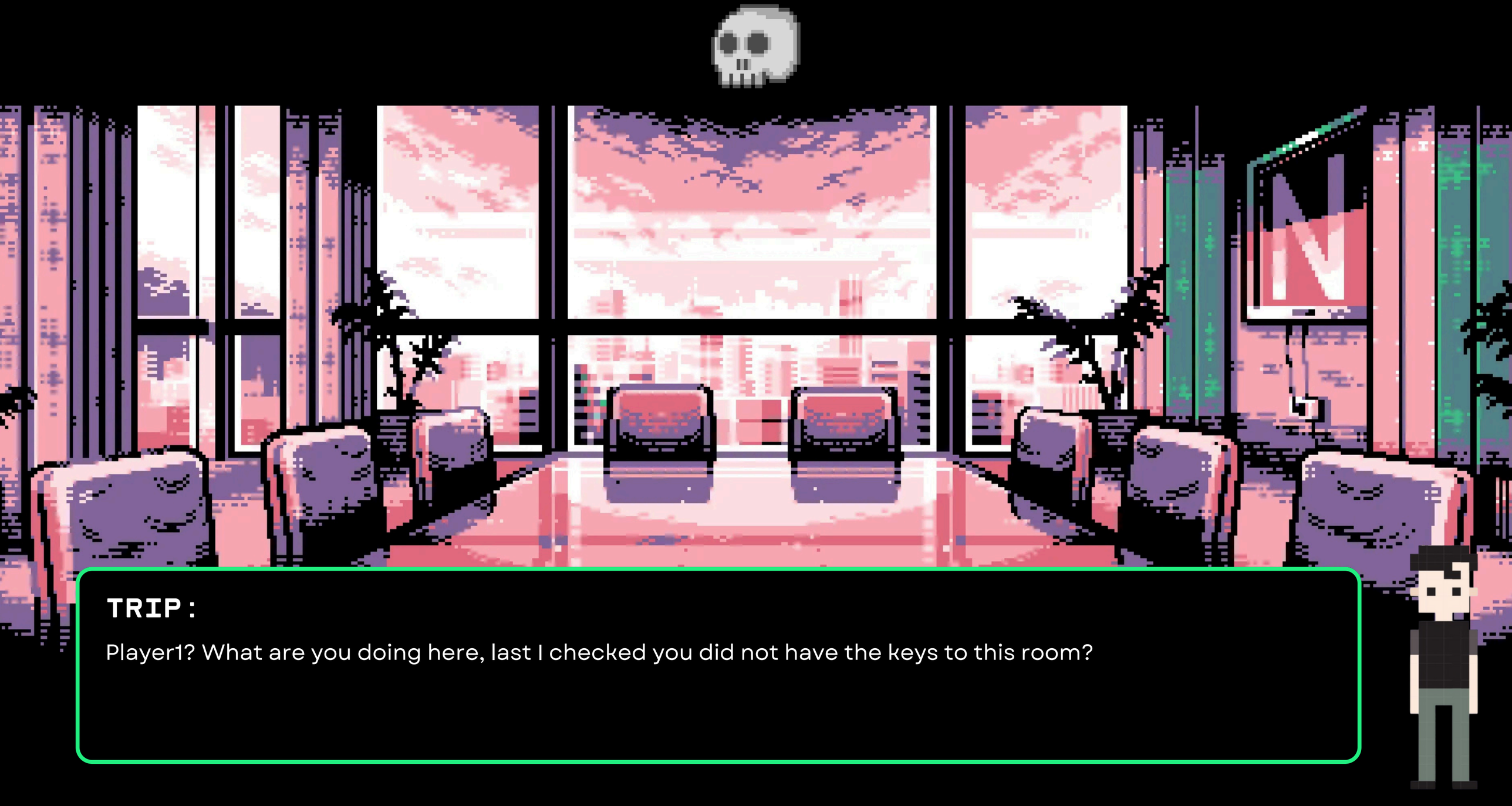

SURE WE CAN DO THAT
SORRY, I DON'T KNOW HOW TO DO THAT
YOU :
I hate my life

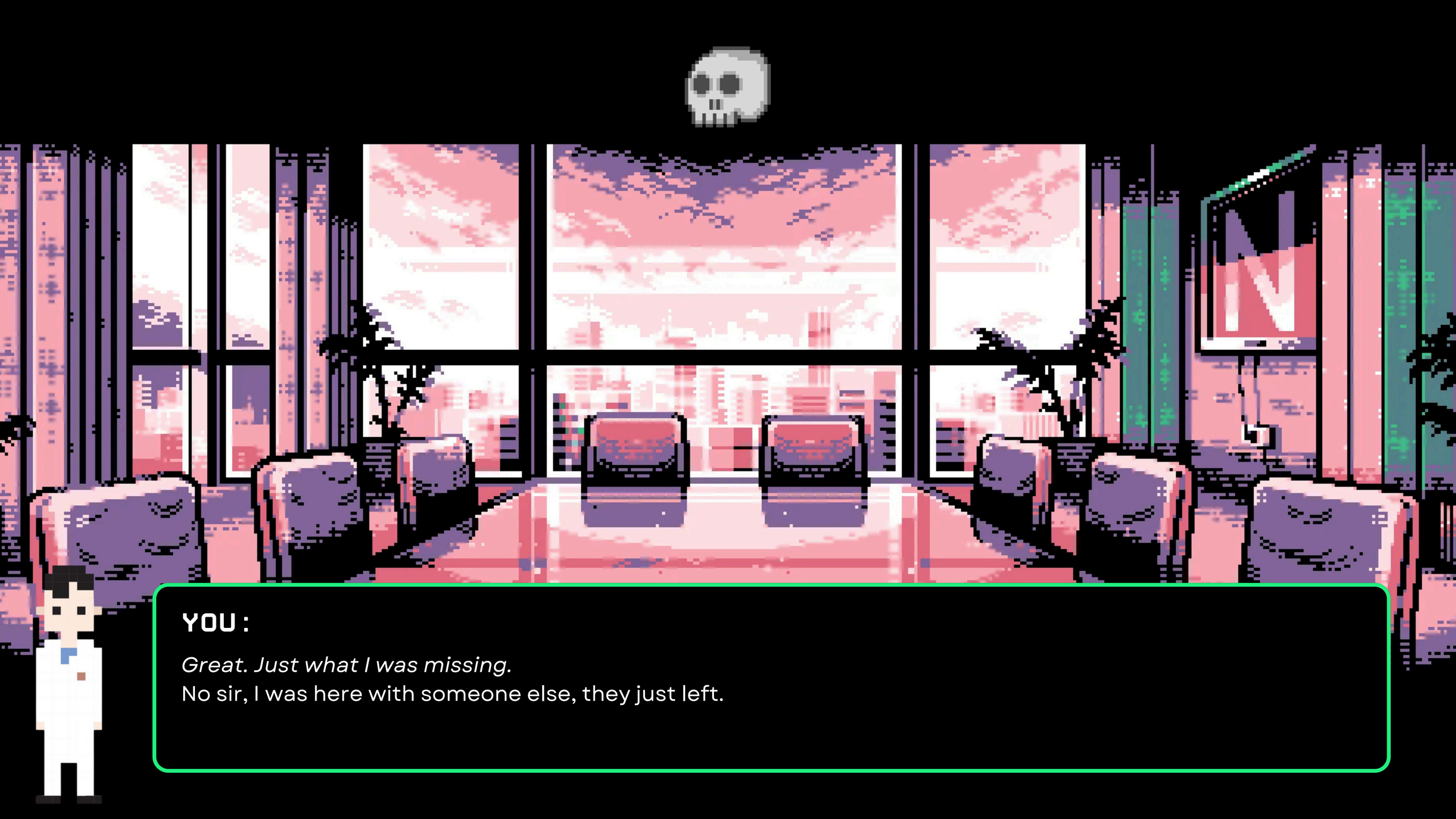

YOU :
I’m sorry, I do not know how to do that T_T
Feels like deja vu

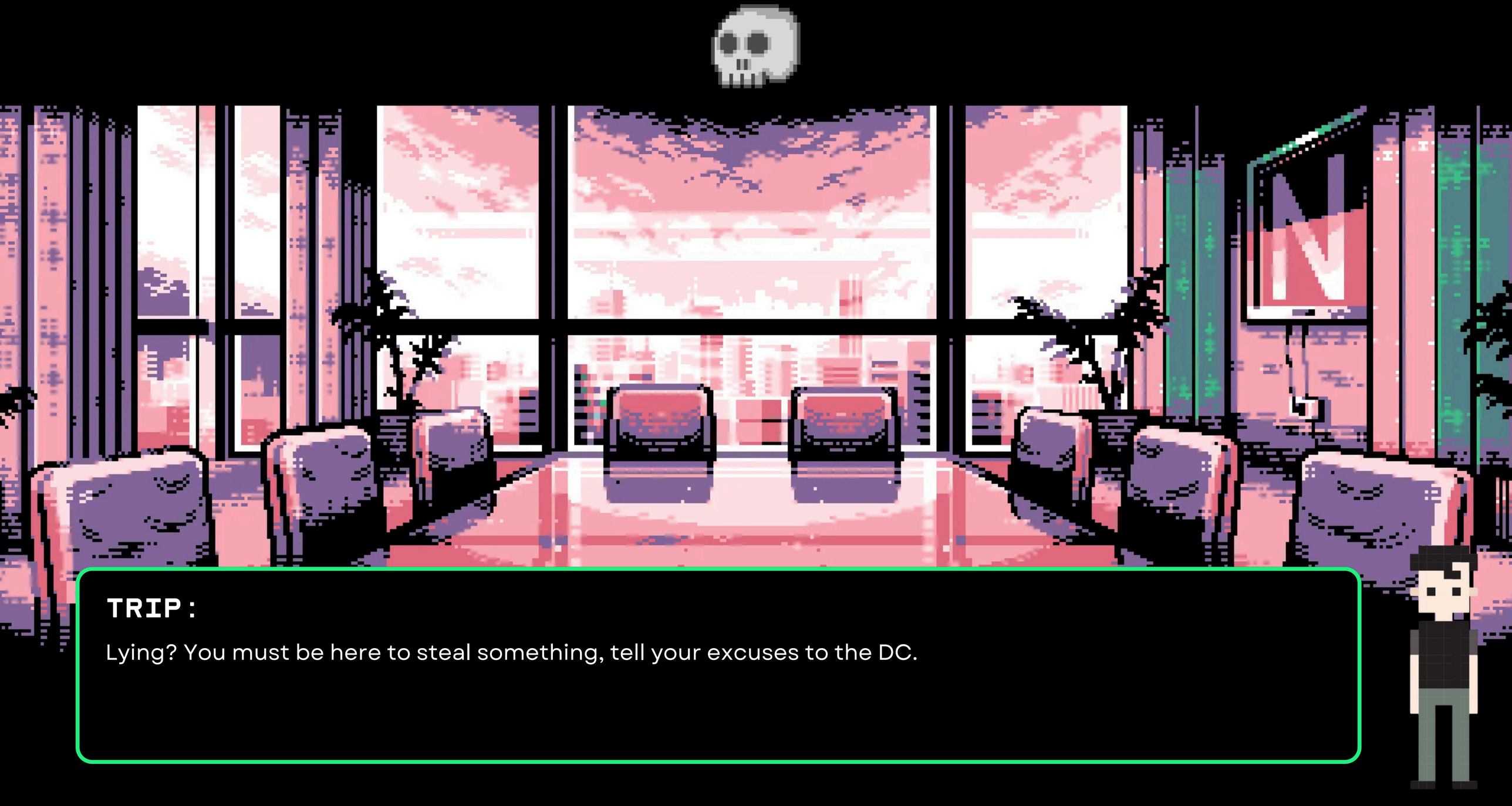

ITISMA CRUSH:
Oh, no problem. Guess then we can leave, could you please close up the meeting room behind me?

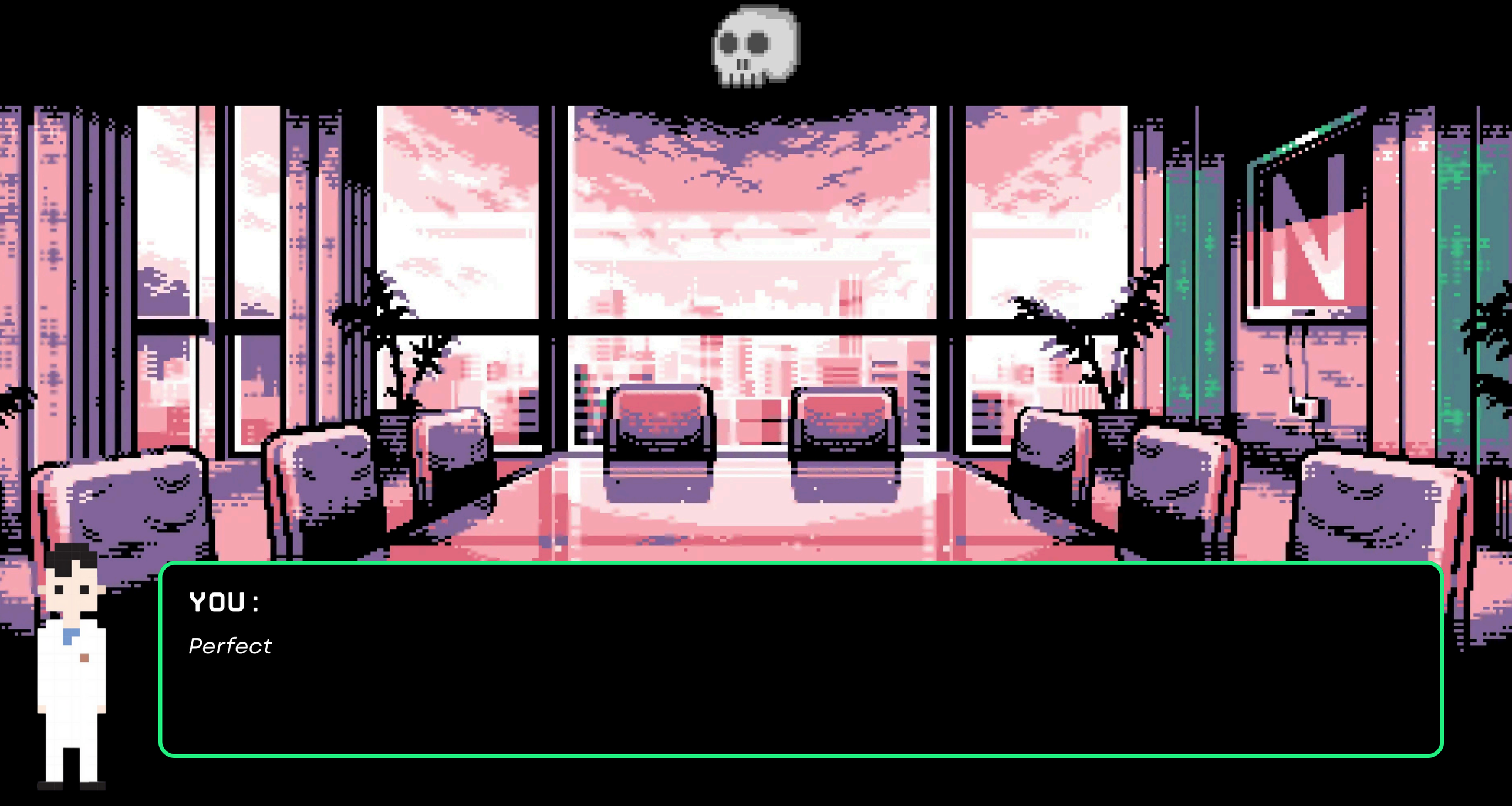

YOU :
Sure
Of course this happens


ITISMA LEAVES
PROFESSOR TRIP ENTERS


TRIP :
Player1? What are you doing here, last I checked you did not have the keys to this room?

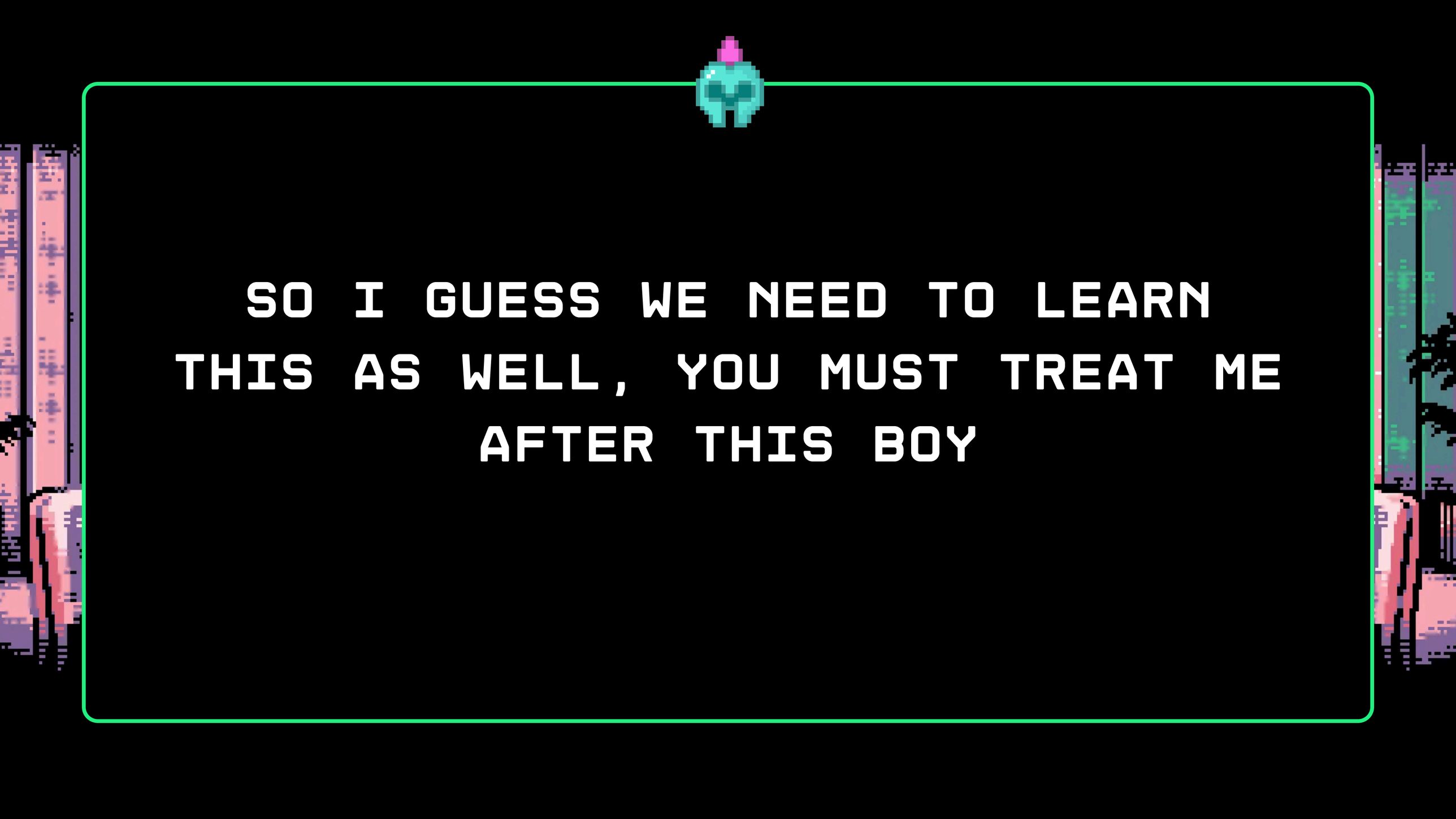

YOU :
Great. Just what I was missing.
No sir, I was here with someone else, they just left.


TRIP :
Lying? You must be here to steal something, tell your excuses to the DC.


YOU :
Perfect

GAME OVER

GO TO LAST CHECKPOINT

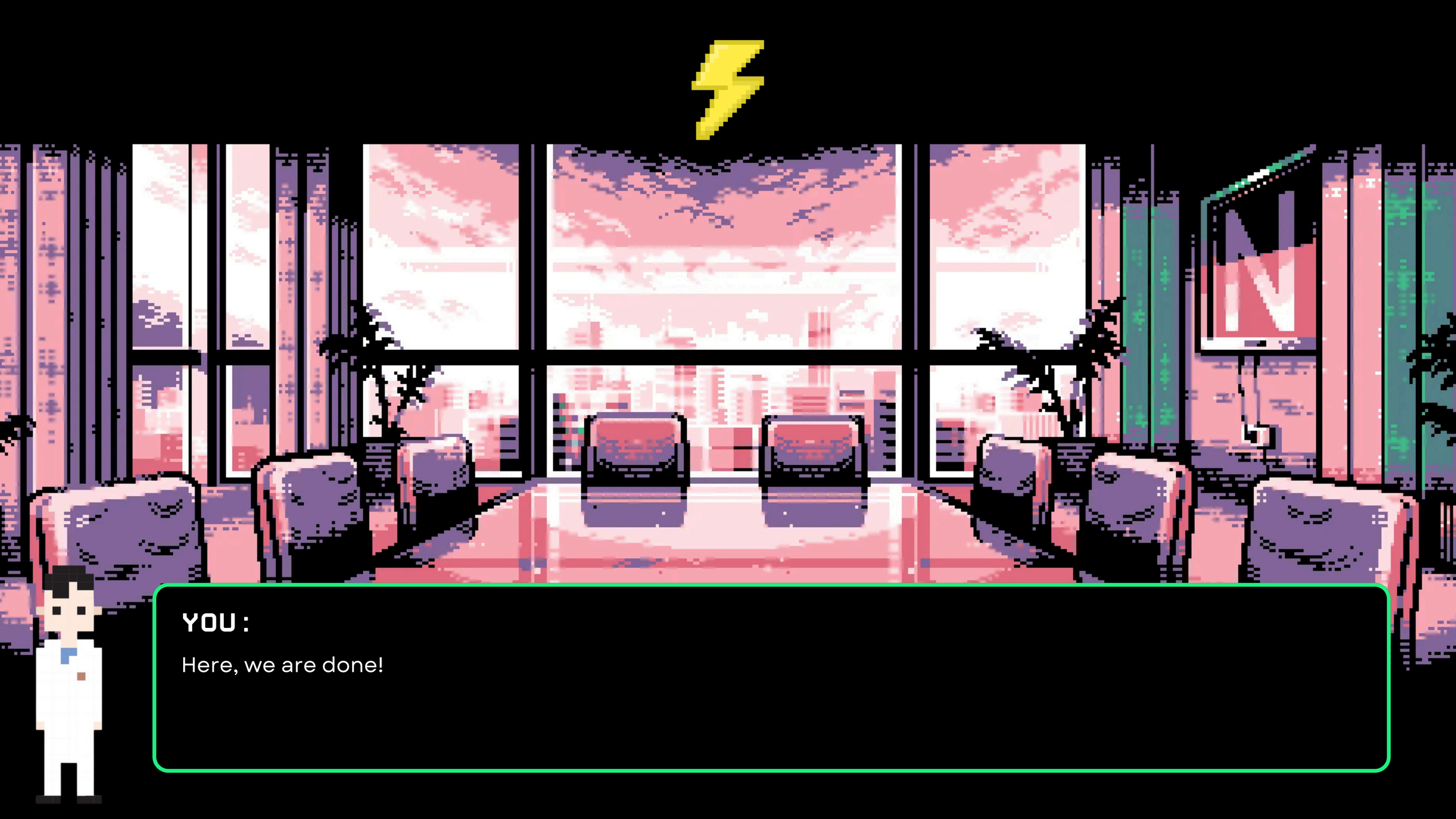

SO I GUESS WE NEED TO LEARN
THIS AS WELL, YOU MUST TREAT ME
AFTER THIS BOY

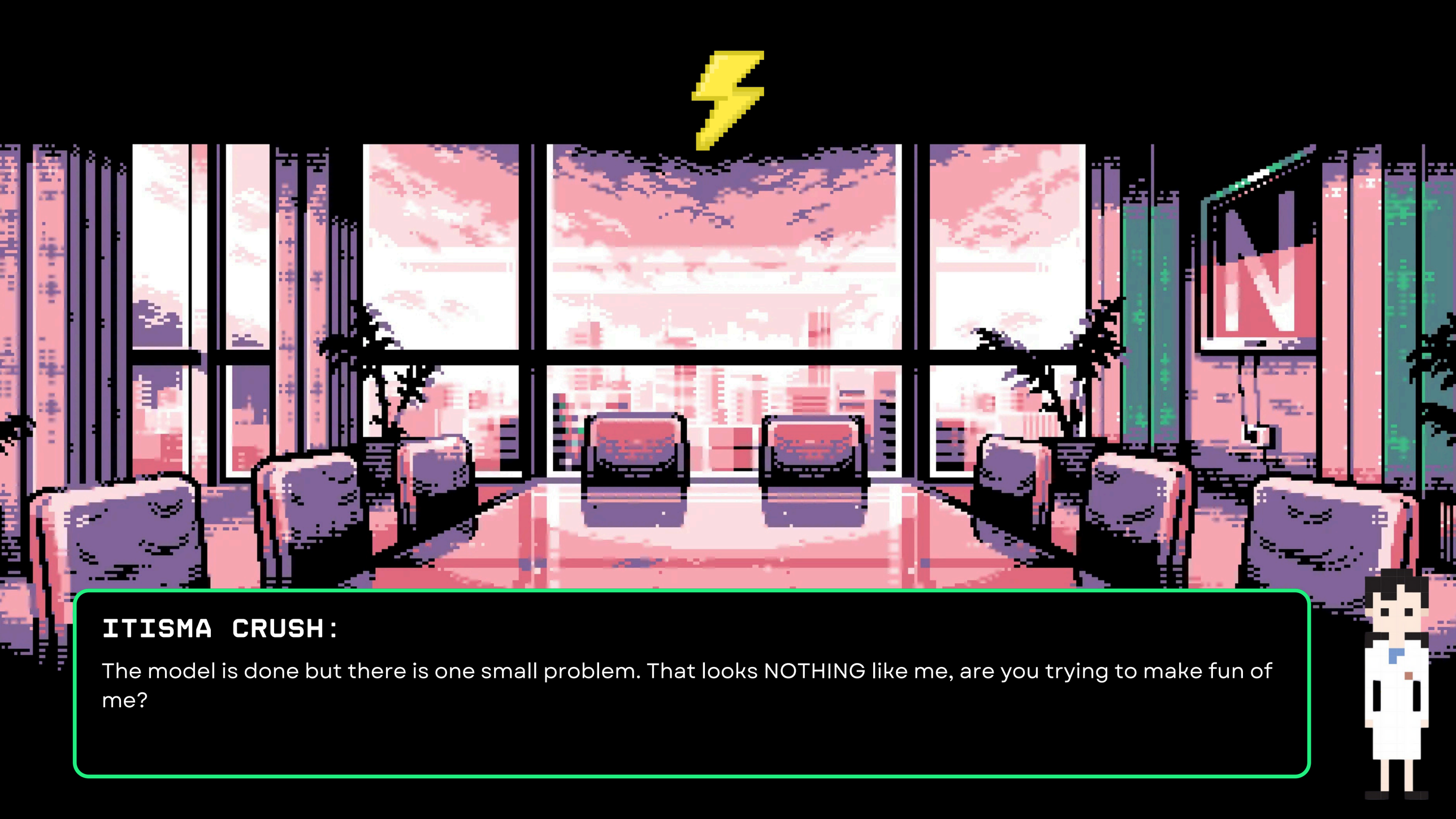
UNDERSTANDING OBJECTS

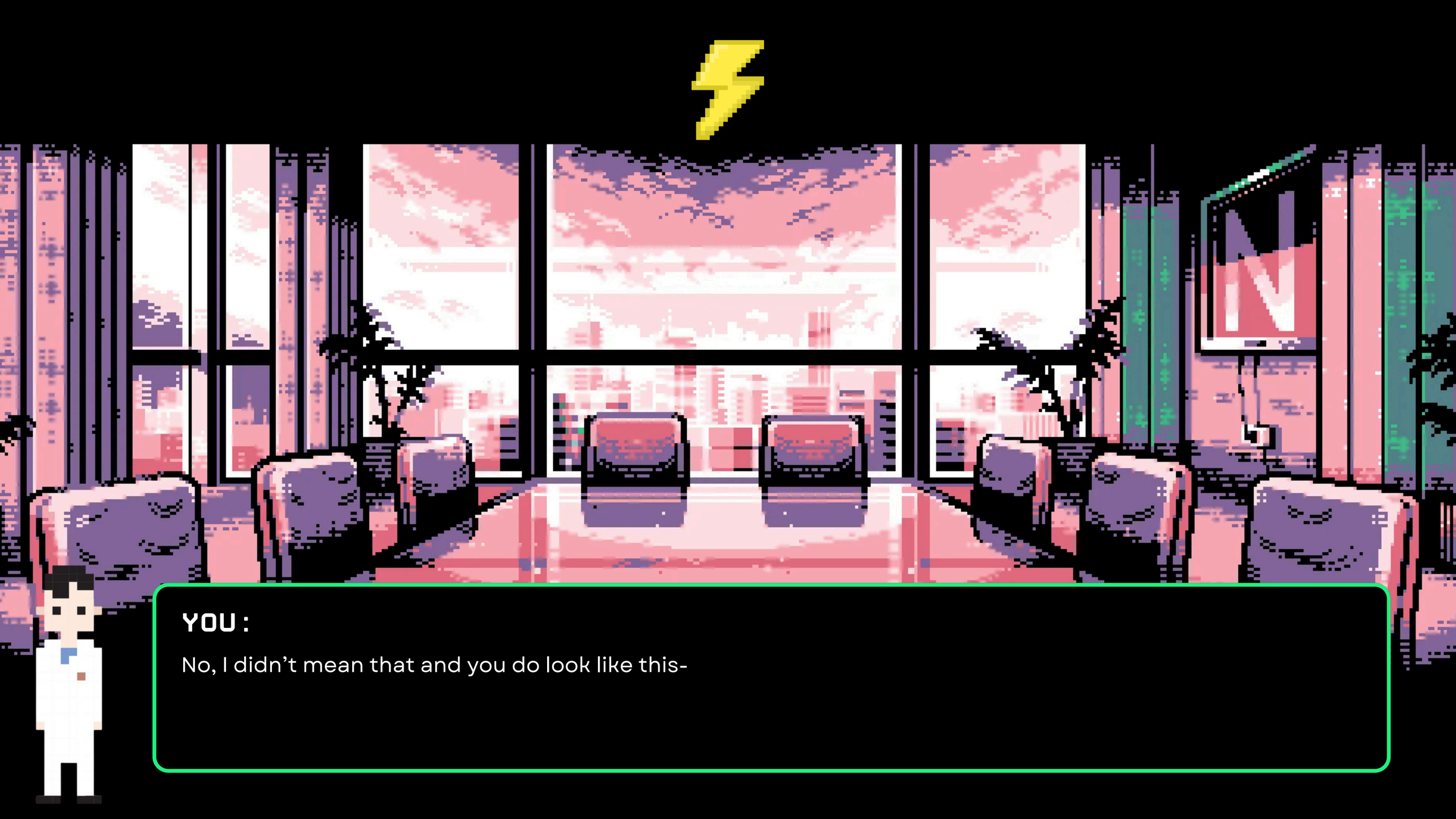
- Objects are stored in the form of .obj files
- These objects have five sections:
  a. usemtl and mtllib describe the look of the model. We won't use this in this tutorial.
  b. v is a vertex
  c. vt is the texture coordinate of one vertex
  d. vn is the normal of one vertex
  e. f is a face
- v, vt and vn are followed by the coordinates required.
- As for f, for a sample value of "f 8/11/7 7/12/7 6/10/7",
  - The three values correspond to the three vertices of the triangle
  - 8 = 8th coordinator
  - 11 = 11th texture parameter

READING THE FILE

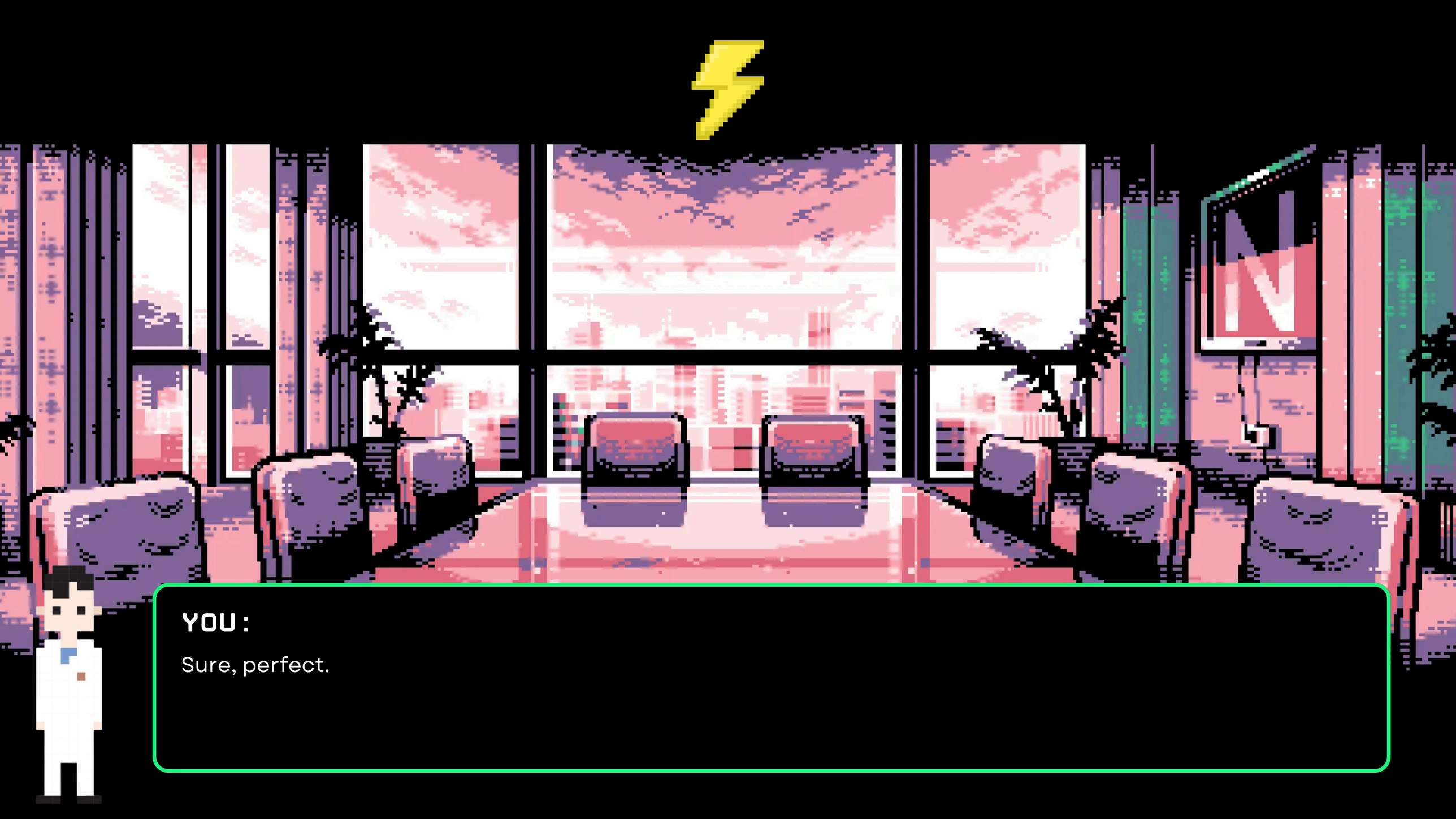
- In simple words, we read the file line by line, separate content on each line and add it to the respective list to store them.
- We then plot them like we usually plot points and add textures to them.
- We add a condition for the first element from the split to separate the different values.
- Then we pass the respective vectors to the plot function rather than our arrays.

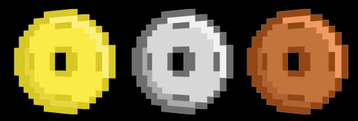

YOU :
Here, we are done!


ITISMA CRUSH:
The model is done but there is one small problem. That looks NOTHING like me, are you trying to make fun of me?


YOU :
No, I didn’t mean that and you do look like this-


ITISMA CRUSH:
Alright then, ThAnK yOu FoR yOuR hElP (leaves)


YOU :
Sure, perfect.

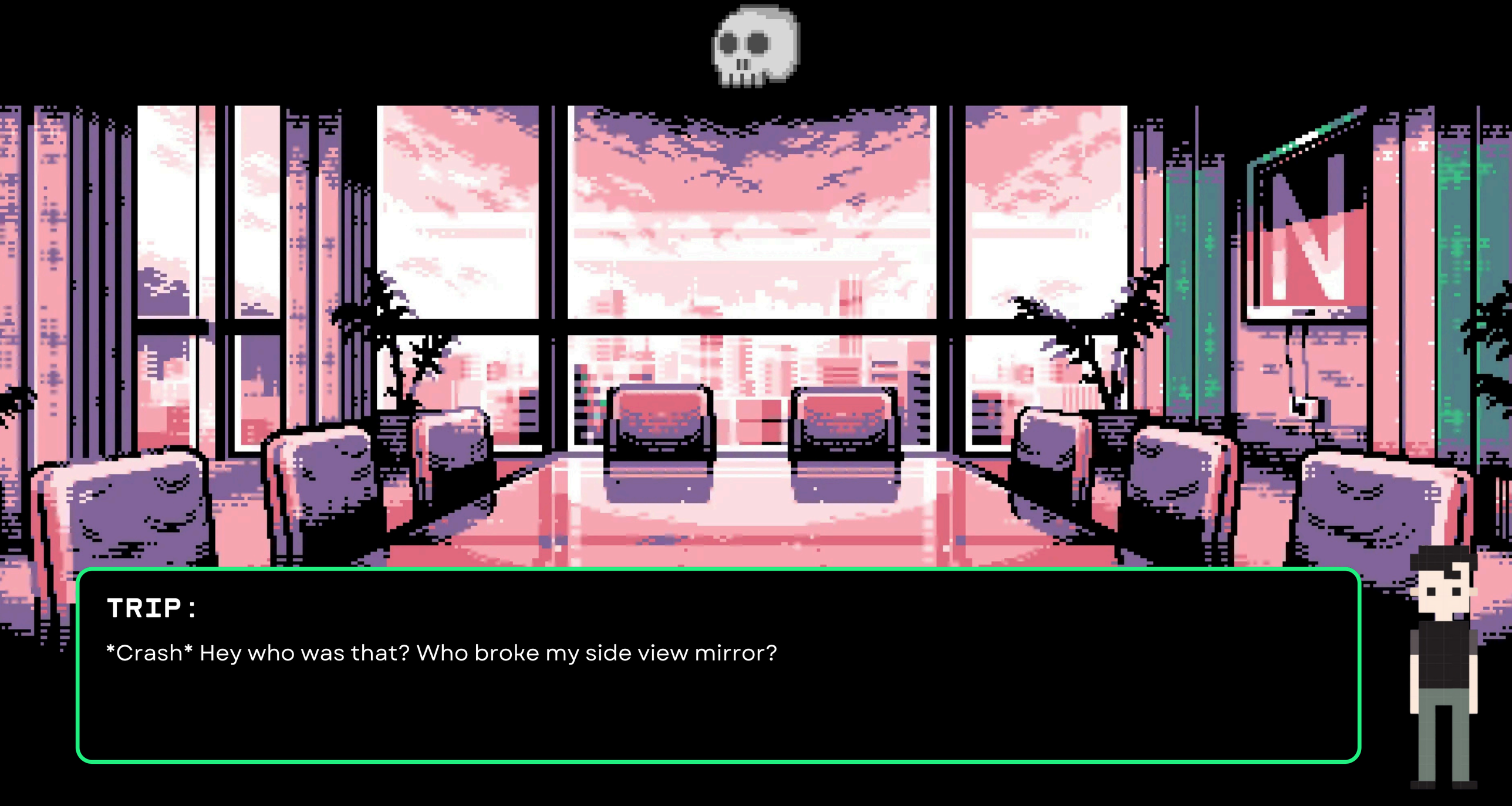


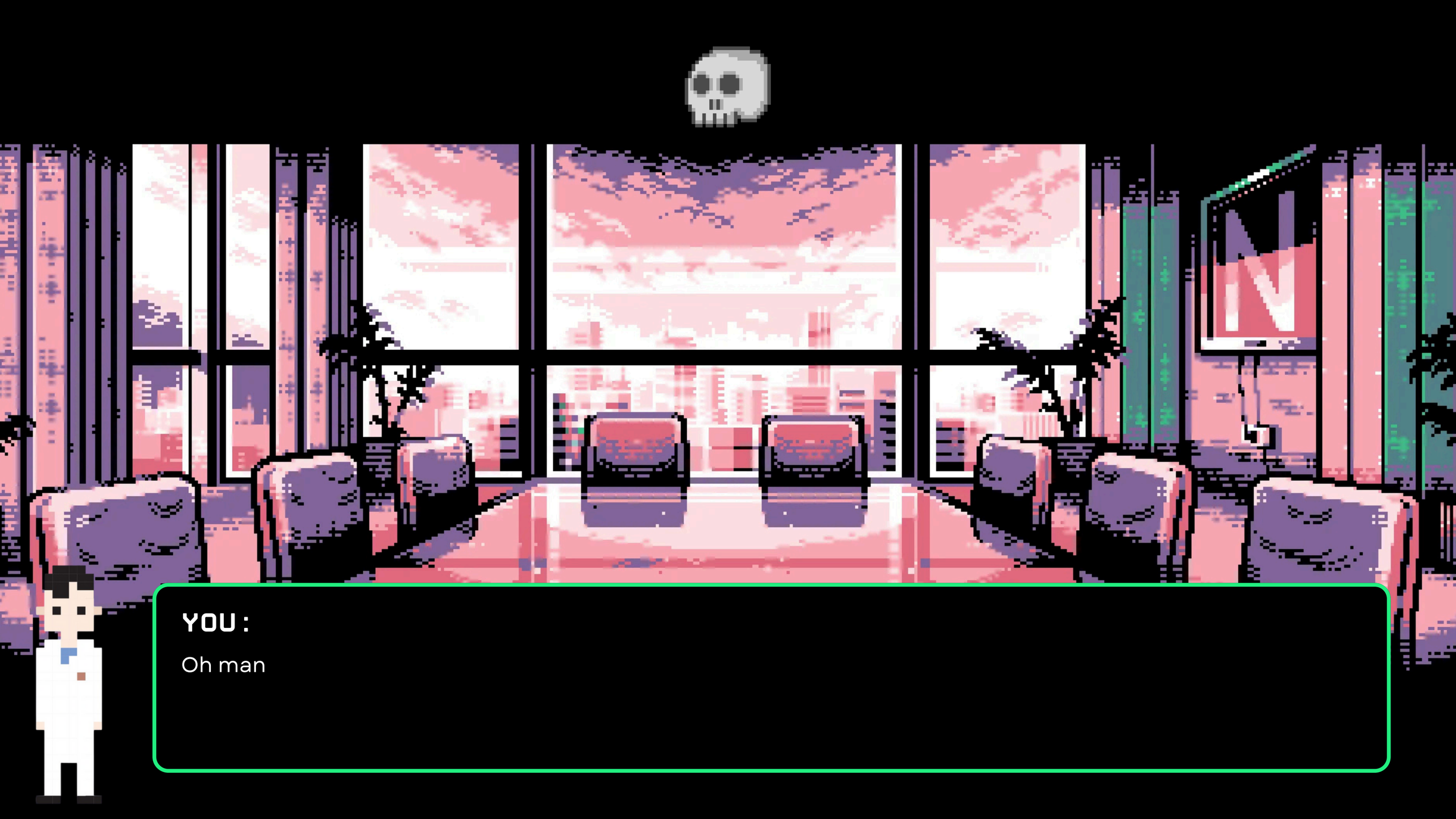
NOW YOU KNOW ABOUT HOW TO LOAD OBJ MODELS INTO OPENGL! HAVE FUN LOL

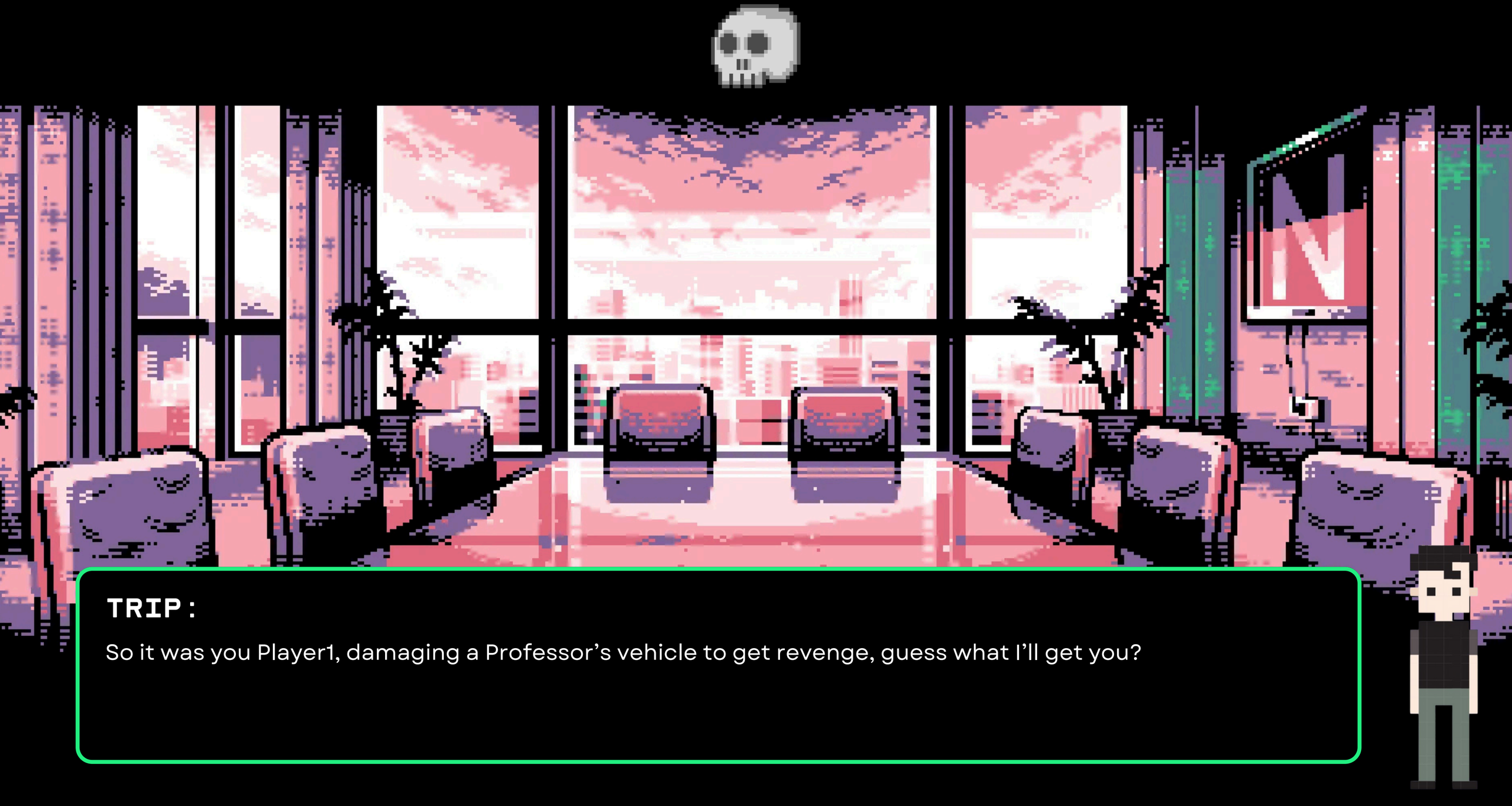
The obj models can be created using 3D modelling software like Blender. The main reason to use them is to avoid manually typing every single point every time we write a program.

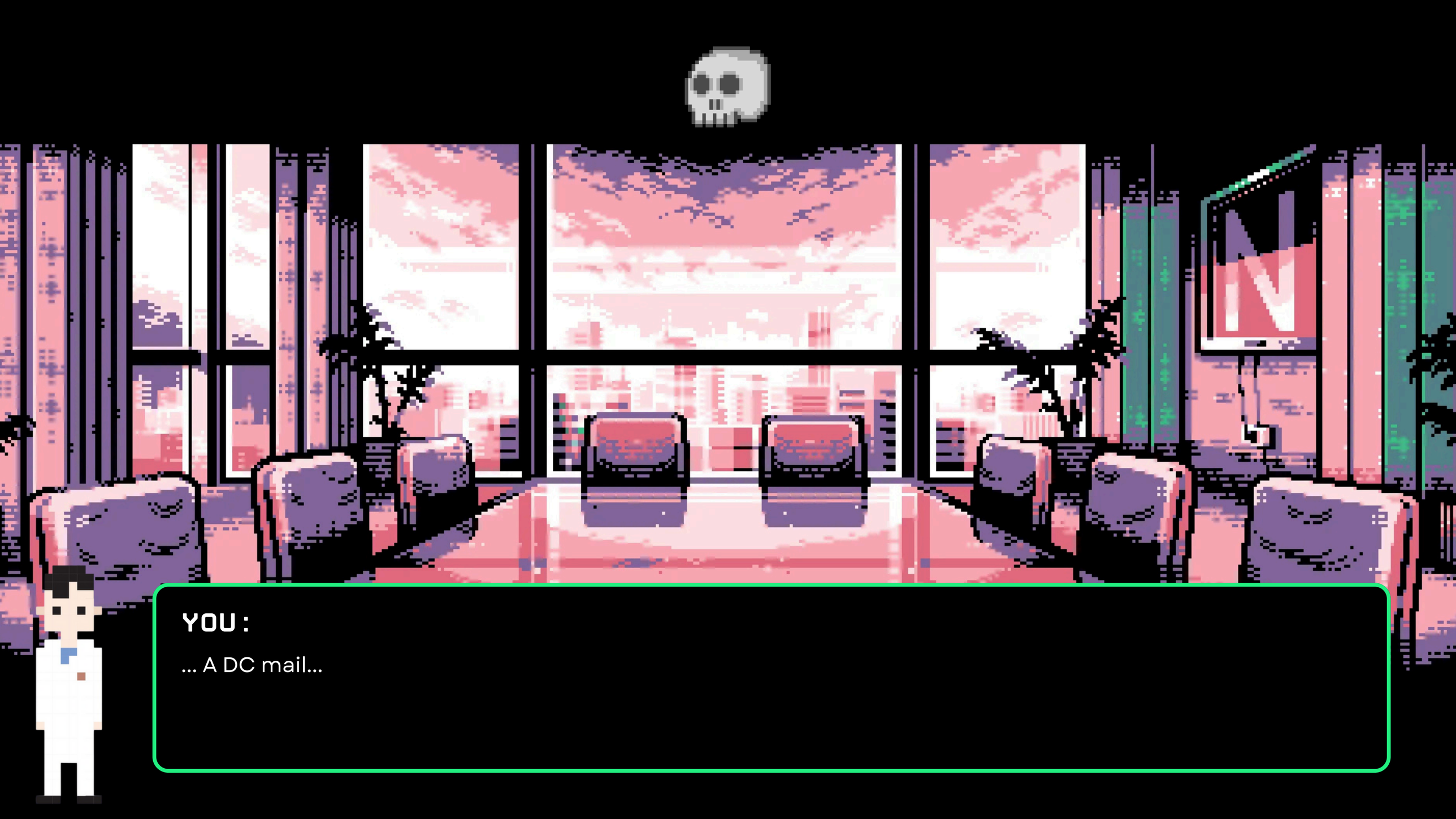

GO TO THE BUS STOP
GO TO THE PARKING LOT
YOU :
Now that that's all over, let’s head home.

YOU GO TO THE PARKING LOT, AND SOME KIDS ARE PLAYING CRICKET YOU ASK THEM AND THEY LET YOU PLAY FOR ONE OVER, YOU HIT IT HARD AND ITS IN THE AIR


TRIP :
Crash Hey who was that? Who broke my side view mirror?


YOU :
Oh man


TRIP :
So it was you Player1, damaging a Professor’s vehicle to get revenge, guess what I’ll get you?

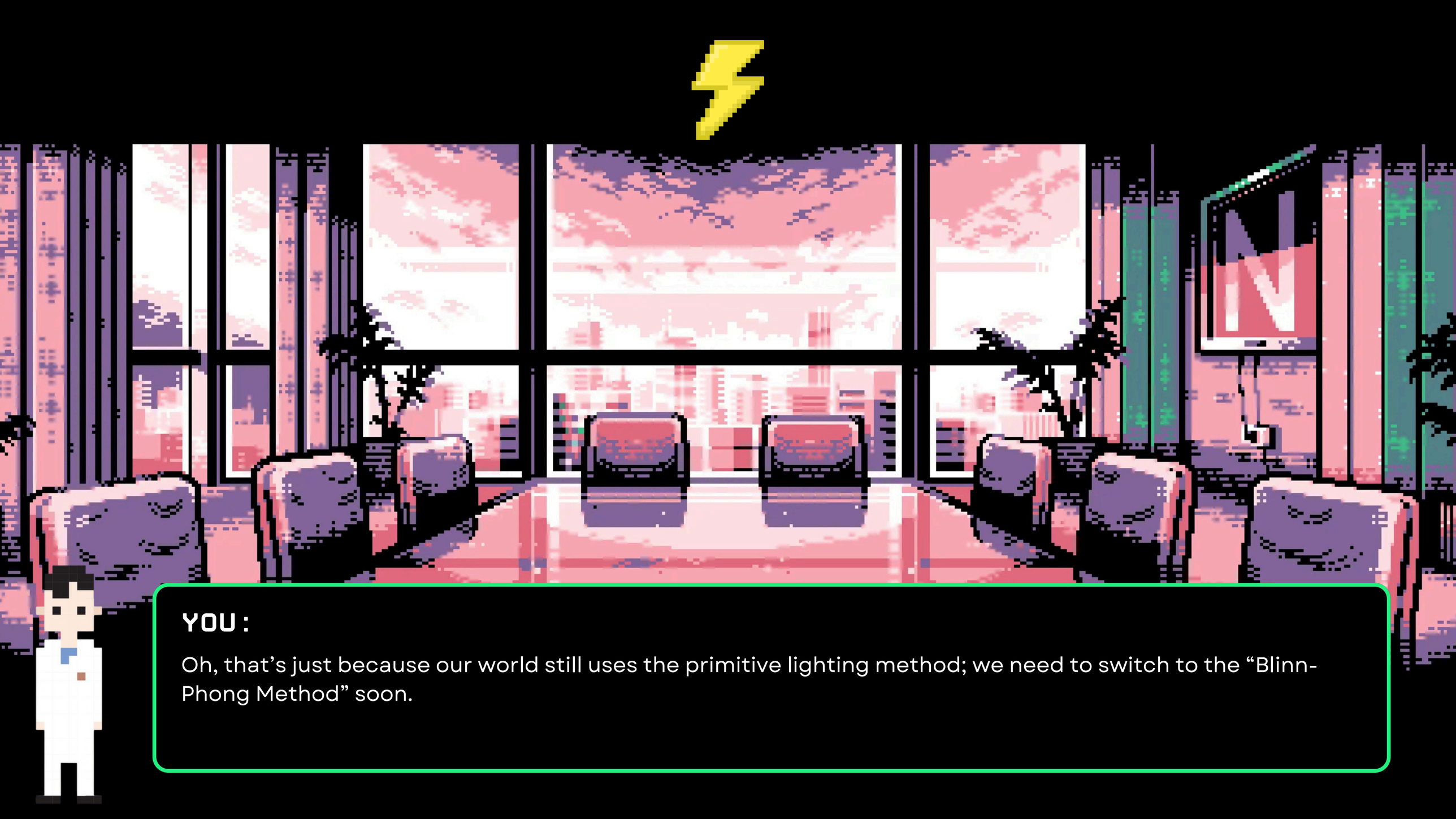

YOU :
... A DC mail...

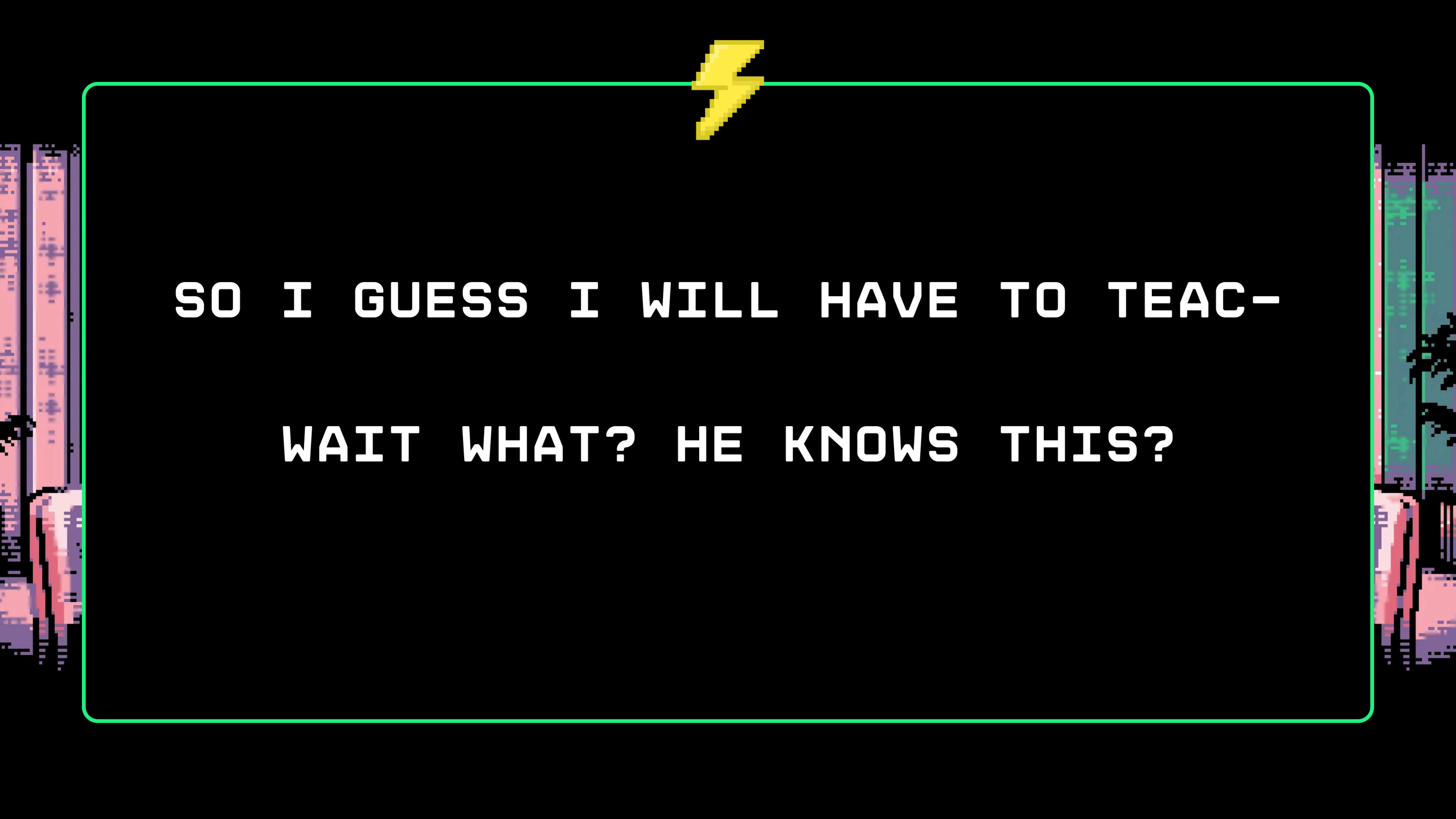
GAME OVER

GO TO LAST CHECKPOINT

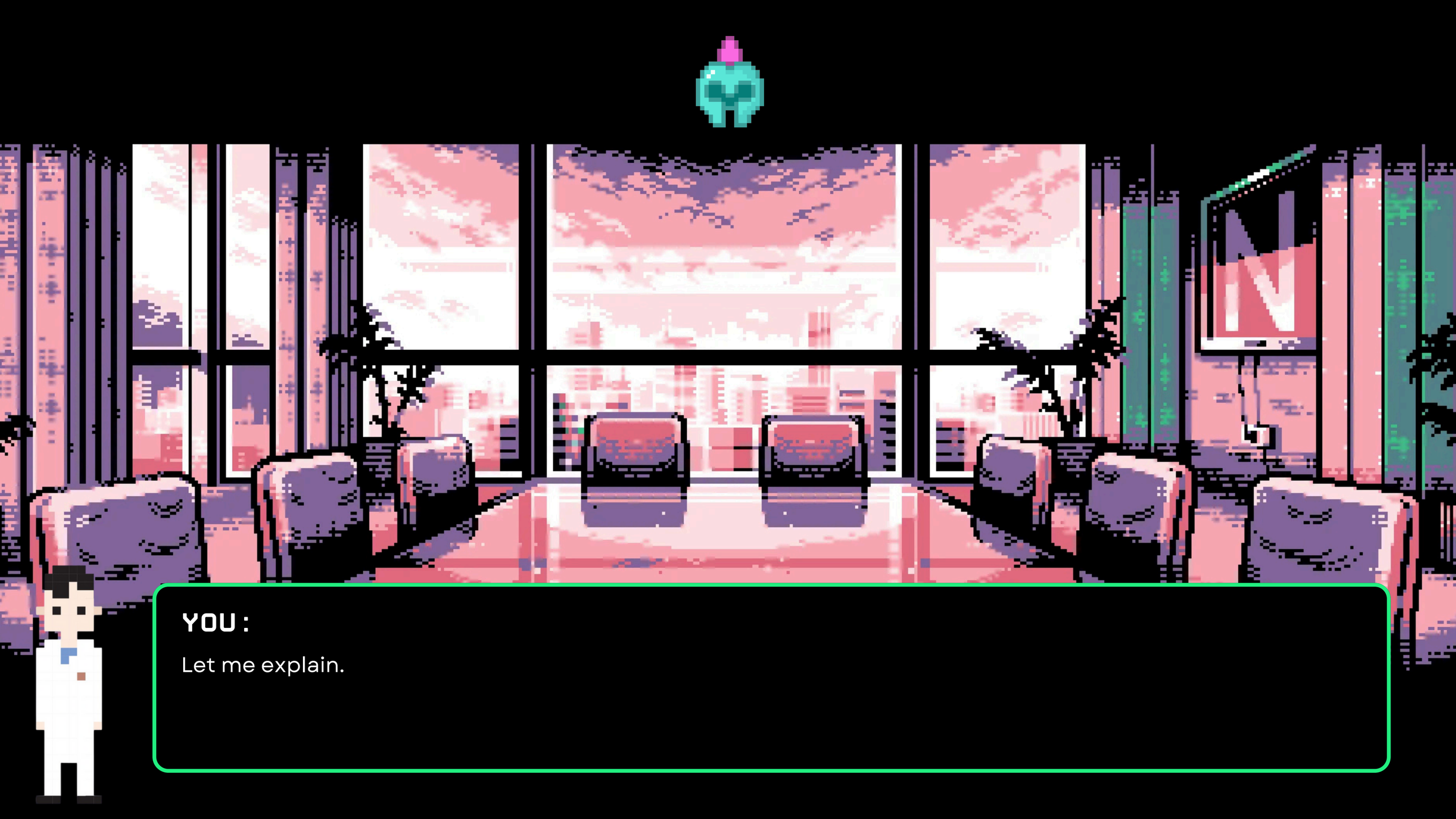

YOU REACH THE BUS STOP


RANDOM STUDENT:
Man, our bus really is old, the colors are way off


YOU :
Oh, that’s just because our world still uses the primitive lighting method; we need to switch to the “Blinn-Phong Method” soon.


SO I GUESS I WILL HAVE TO TEAC-
WAIT WHAT? HE KNOWS THIS?


RANDOM STUDENT:
What is that?


YOU :
Let me explain.

PRIMITIVE LIGHTING

And then god said, “Let there be light”

- Primitive Lighting is the method where the actual observed color of an object is defined by simply multiplying the values of the light source intensities and the actual color of the object.
- This gives a good enough approximation of the light but is extremely color inaccuar

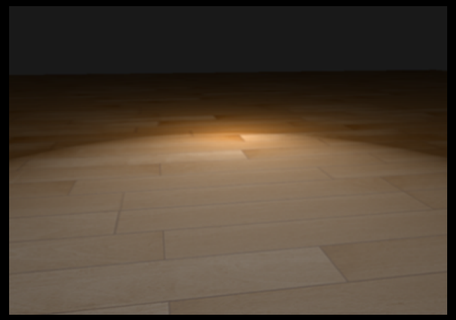
ADVANCED LIGHTING

And then god said, “Let there be light”

- Normally, advanced lighting in OpenGL is done by combining:
  a. Ambient Lighting
  b. Diffuse Lighting
  c. Specular Lighting
- A combination of these lighting methods gives us a pretty accurate depiction of the lighting in real life.
- This is called the ‘Phong’ lighting style.
- The main drawback with this method is that its specular reflections break down in certain conditions, specifically when the shininess property is low resulting in a large (rough) specular area.

ADVANCED LIGHTING

And then god said, “Let there be light”



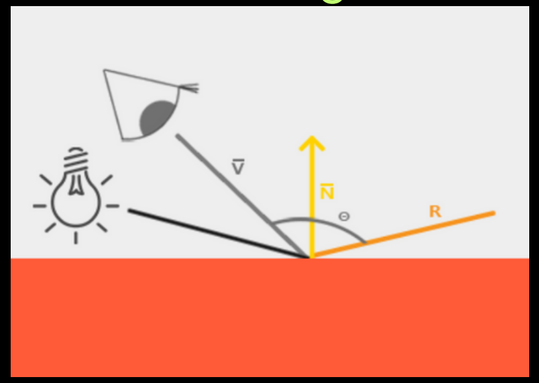
You can see at the edges that the specular area is immediately cut off. The reason this happens is because the angle between the view and reflection vector doesn't go over 90 degrees. If the angle is larger than 90 degrees, the resulting dot product becomes negative and this results in a specular exponent of 0.0

To overcome this problem, we introduce a halfway vector and use the “Blinn-Phong” lighting method

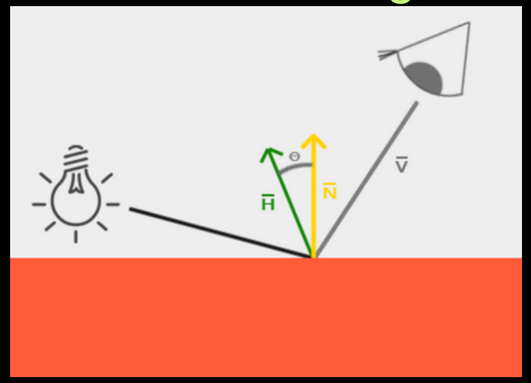
BLINN-PHONG LIGHTING

- In this method, instead of using the angle between the view vector and the normal vector as the Specular metric, we define a vector halfway between the view and the lighting vector and use the angle between the Halfway Vector and the Normal Vector as the Specular metric, ensuring that the angle never exceeds 90°.

Phong



Blinn-Phong



Where,

$$\bar{H} = \frac{\bar{L} + \bar{V}}{||\bar{L} + \bar{V}||}$$

BLINN-PHONG LIGHTING

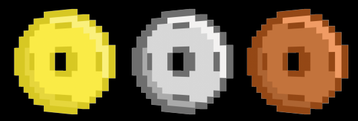
- An example on the previously used image:

Phong

Blinn-Phong


RANDOM STUDENT:
Oh that makes sense, so when are we switching?


YOU :
I don’t know



I DON'T KNOW IF THIS COUNTS AS AN ACHIEVEMENT, BUT OK, NOW YOU KNOW ABOUT ADVANCED LIGHTING IN OPENGL...

These lighting methods prove to be extremely instrumental in various projects as lighting is the most important emphasis-inducing element.
You get off the bus and start walking.

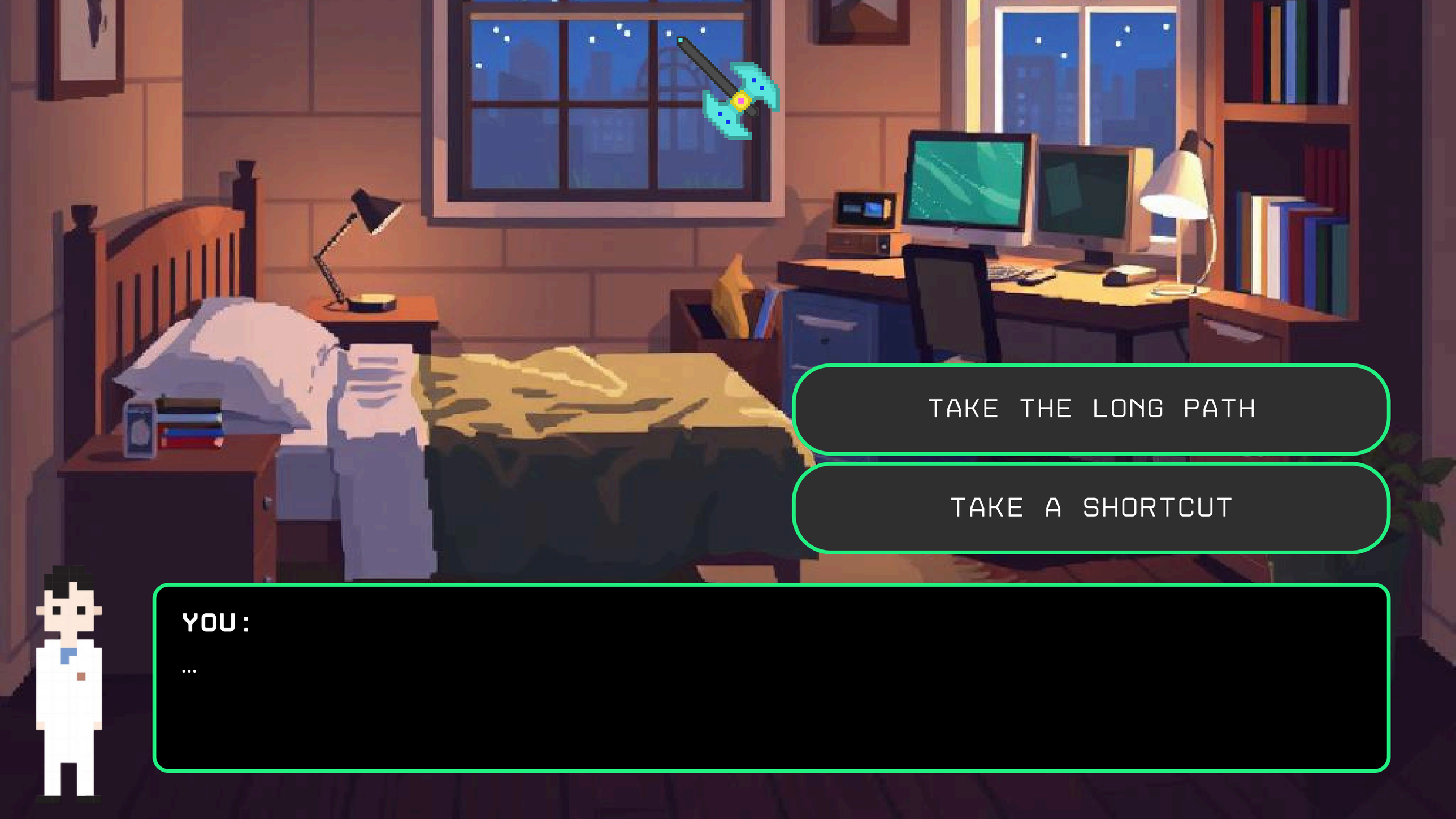
BACK TO LEVELS MENU

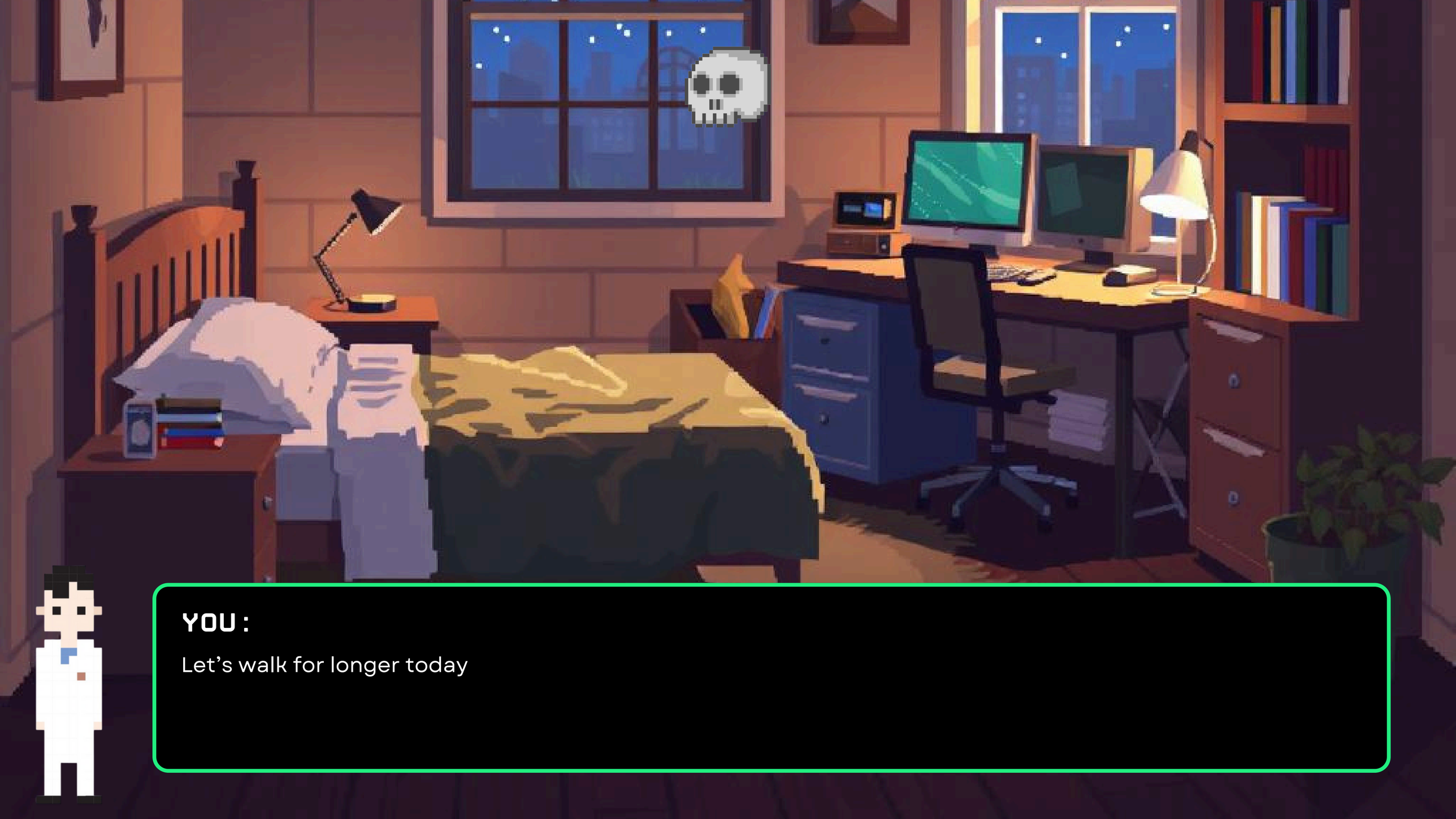
03 18 03

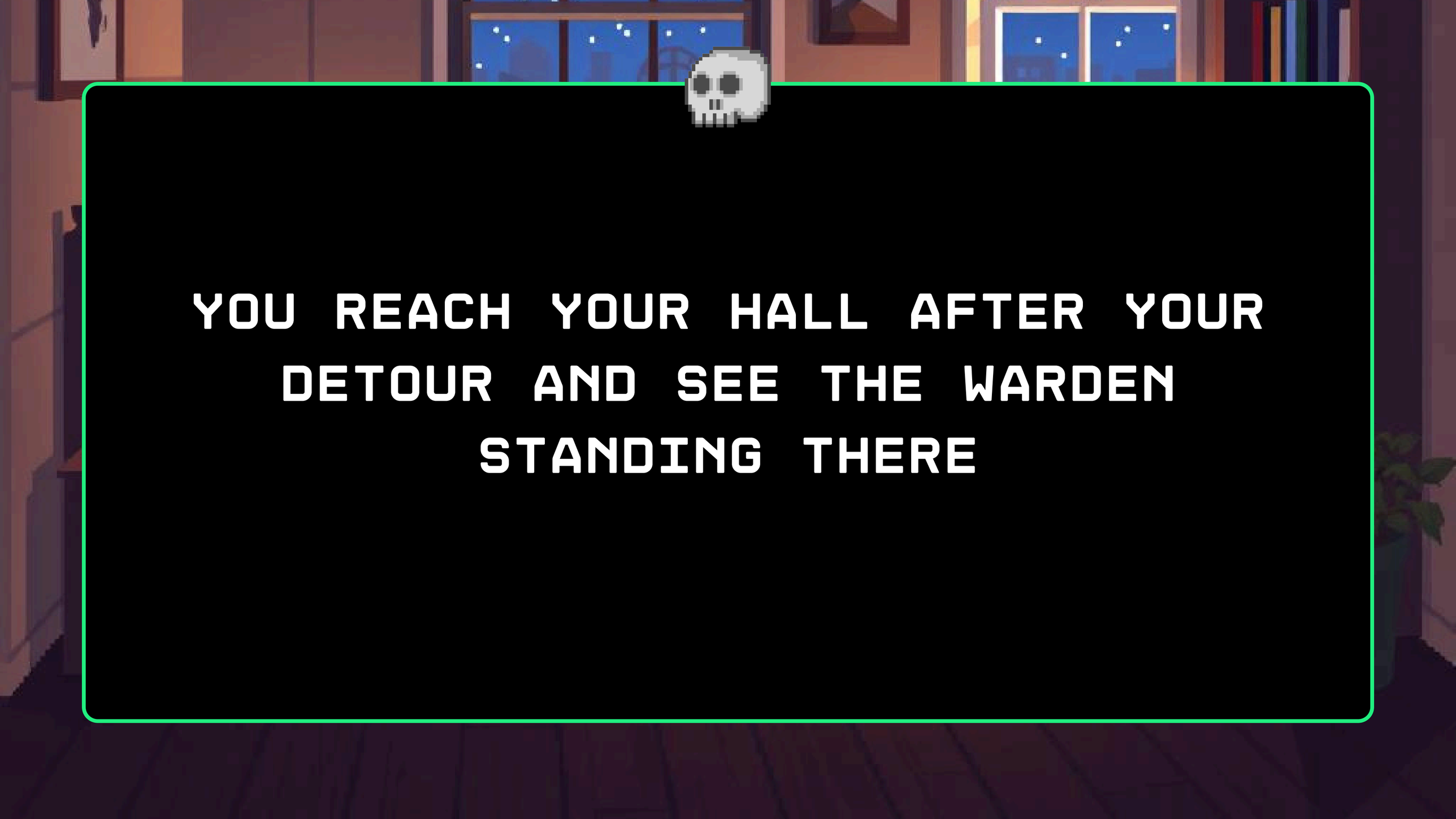
CONGRATULATIONS! YOU HAVE CLEARED THE SOCIETY ARC!!

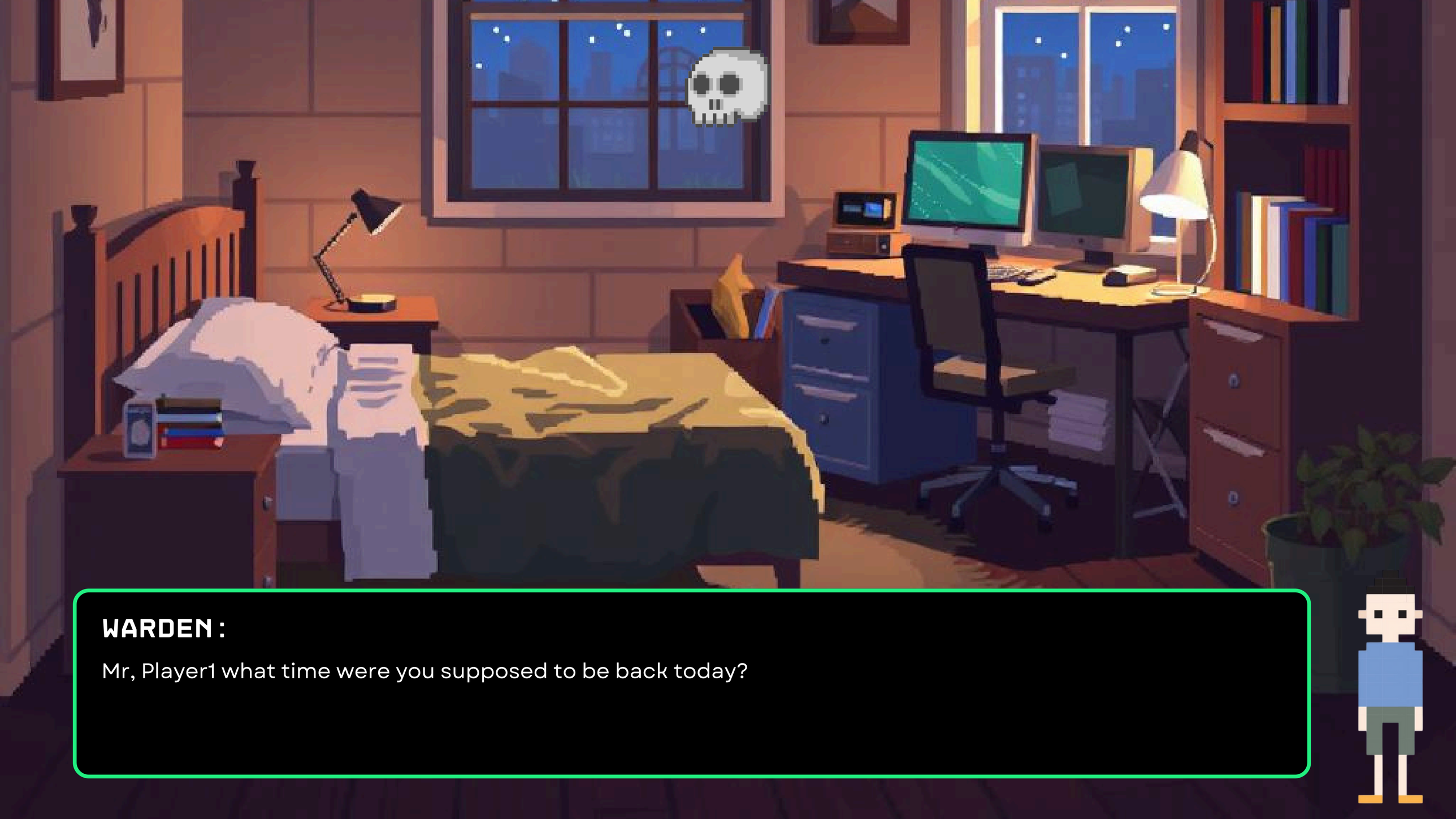
BACK TO LEVELS PAGE

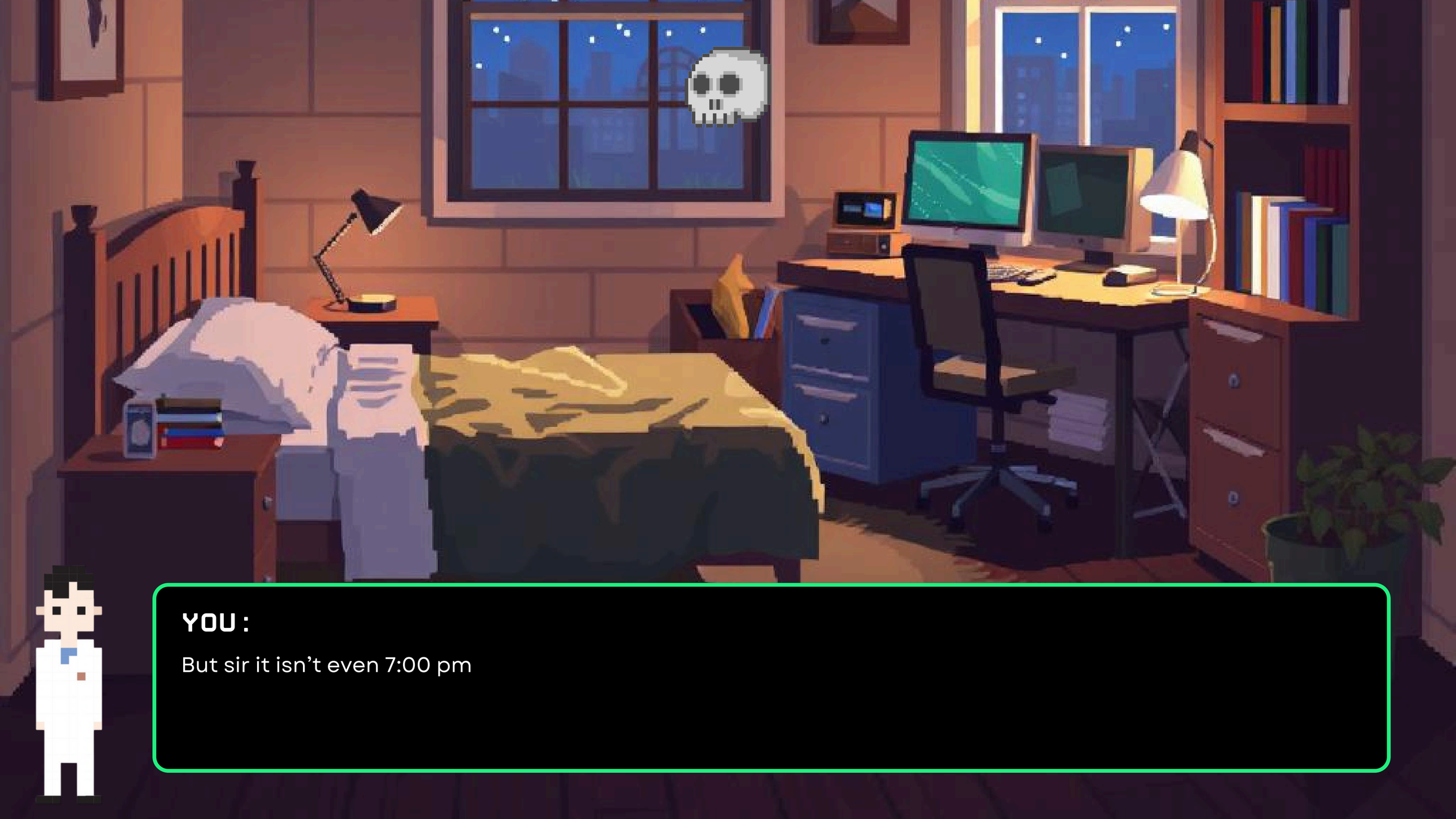
HALL ARC

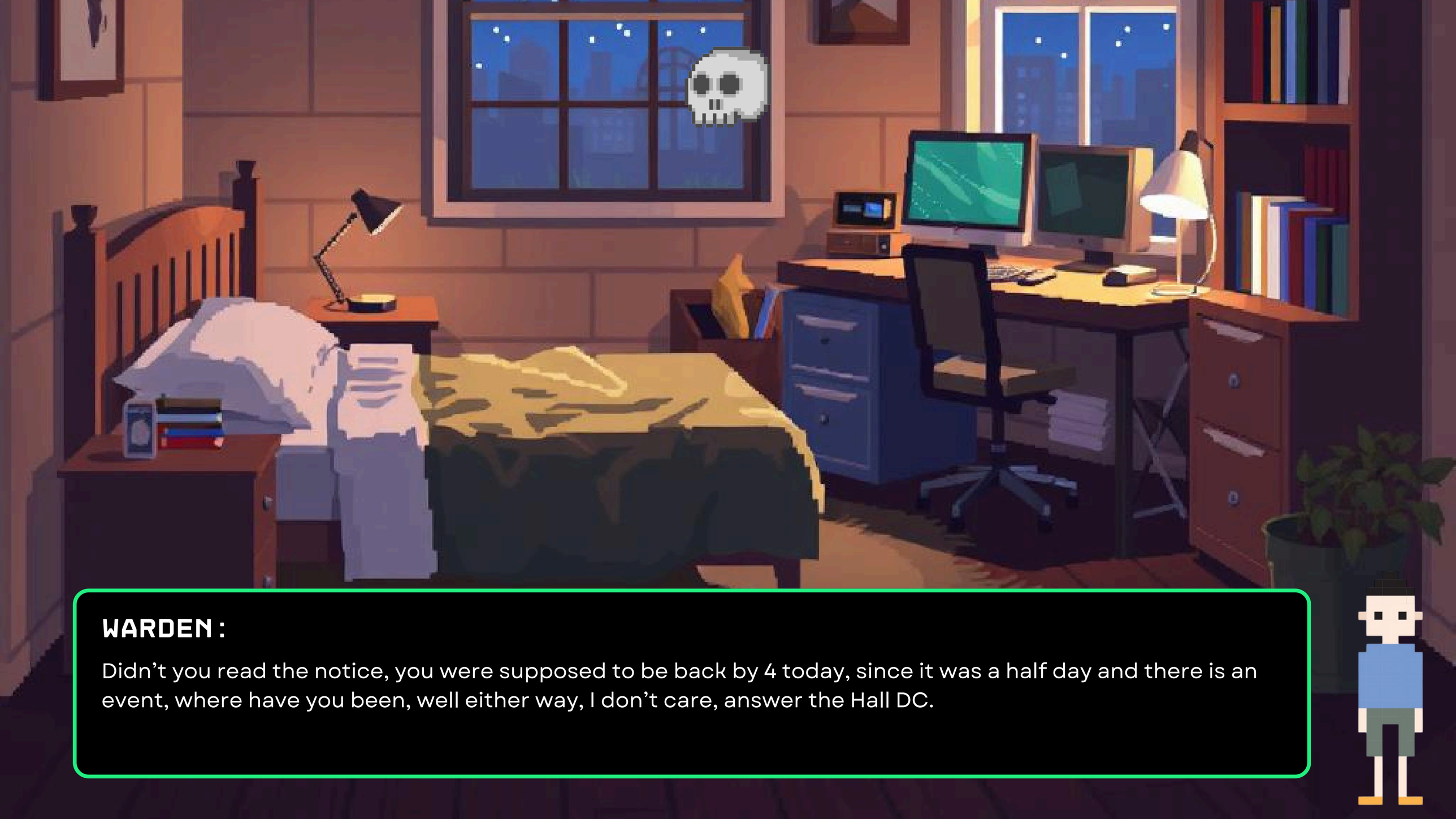

TAKE THE LONG PATH
TAKE A SHORTCUT
YOU :
...


YOU :
Let’s walk for longer today


YOU REACH YOUR HALL AFTER YOUR
DETOUR AND SEE THE WARDEN
STANDING THERE


WARDEN :
Mr, Player1 what time were you supposed to be back today?


YOU :
But sir it isn’t even 7:00 pm

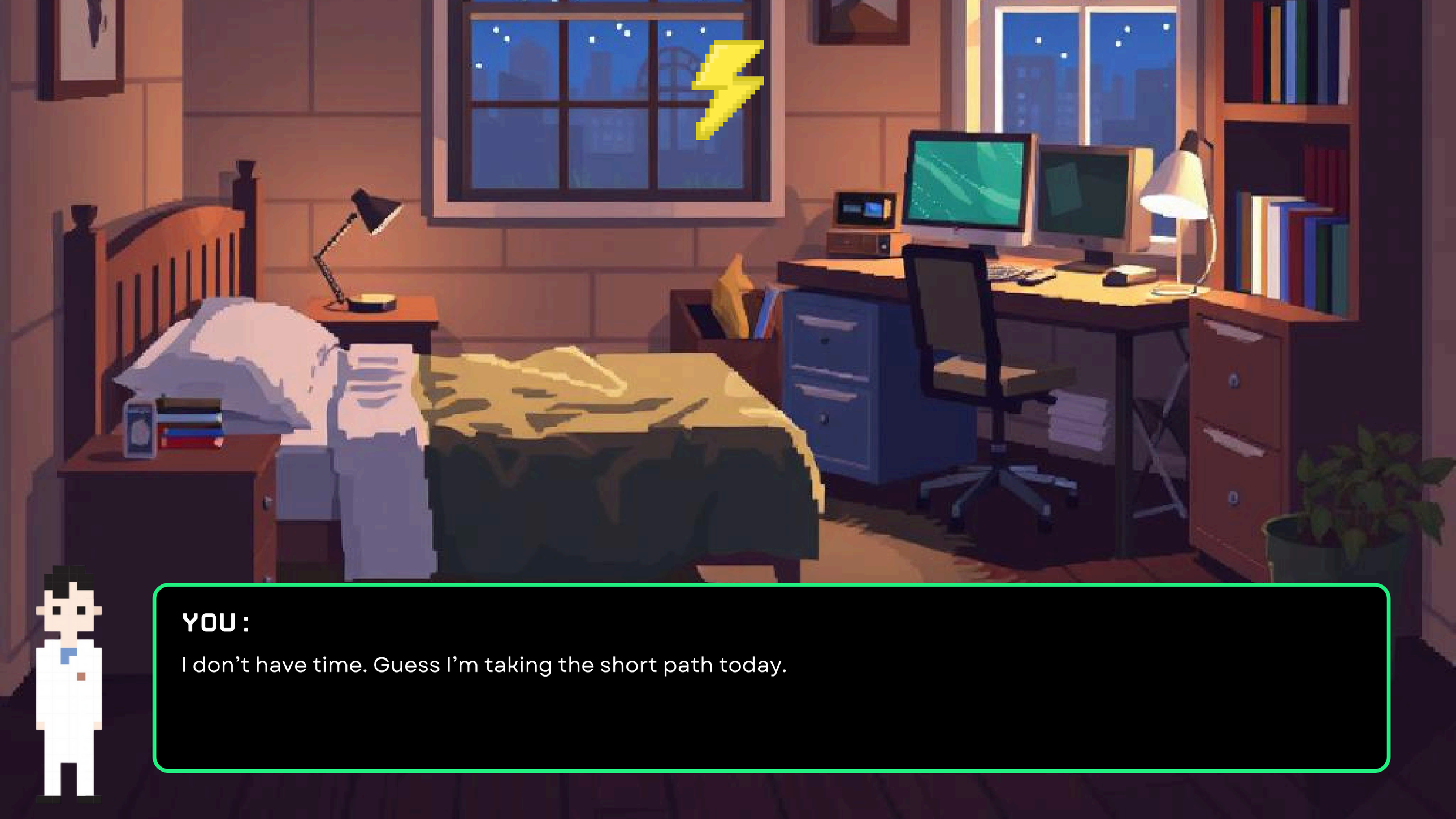


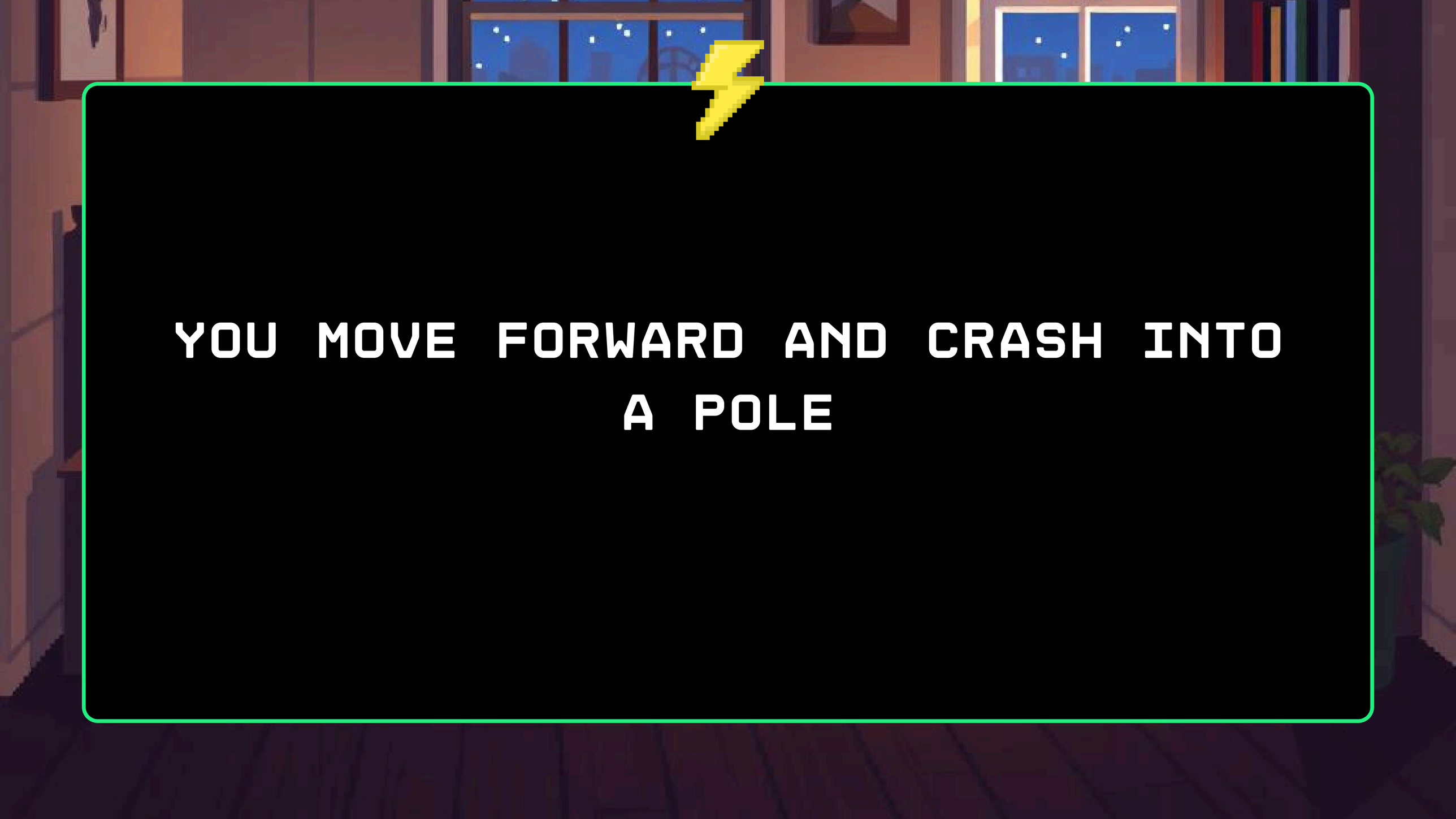
WARDEN :

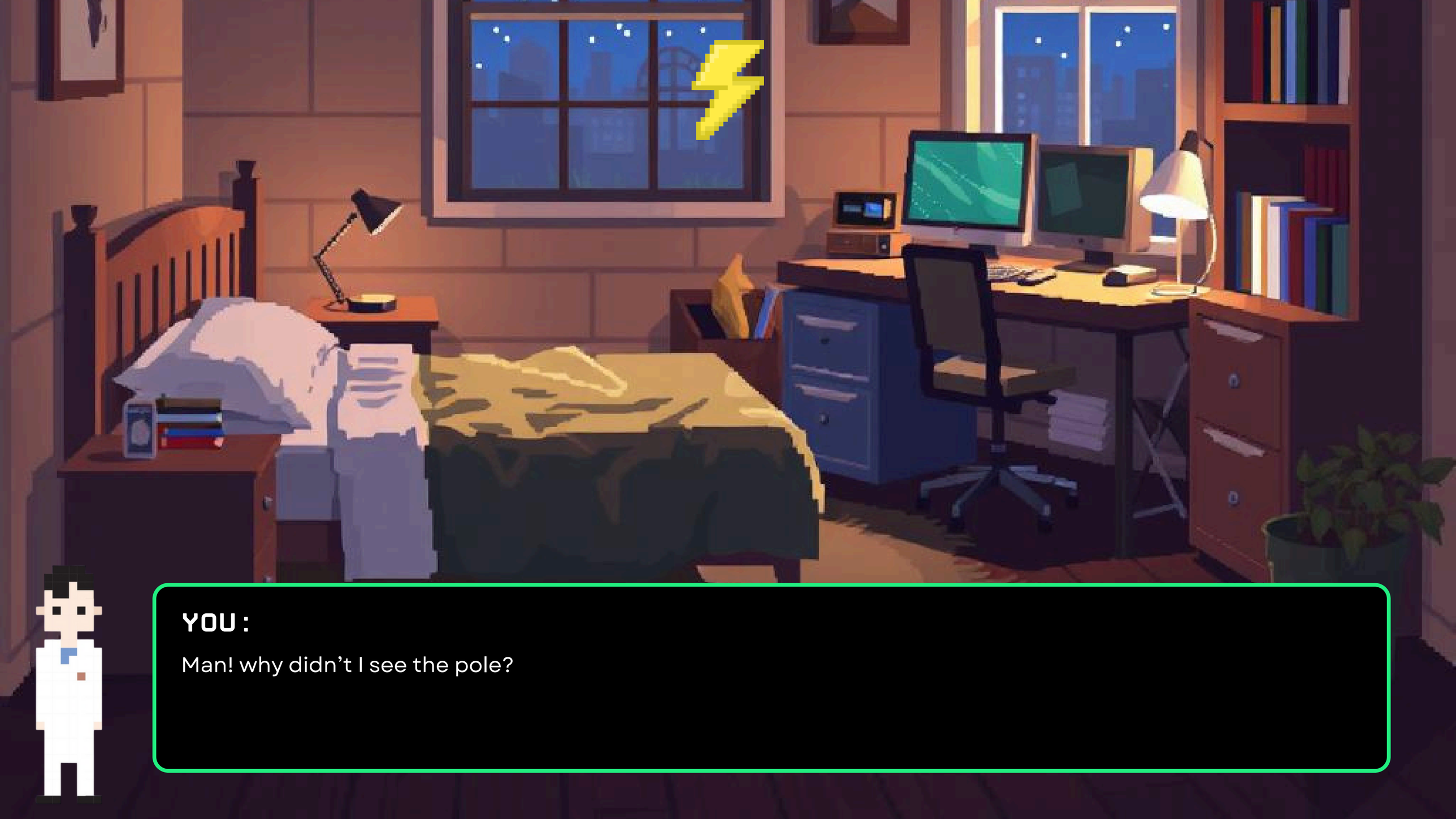
Didn’t you read the notice, you were supposed to be back by 4 today, since it was a half day and there is an event, where have you been, well either way, I don’t care, answer the Hall DC.

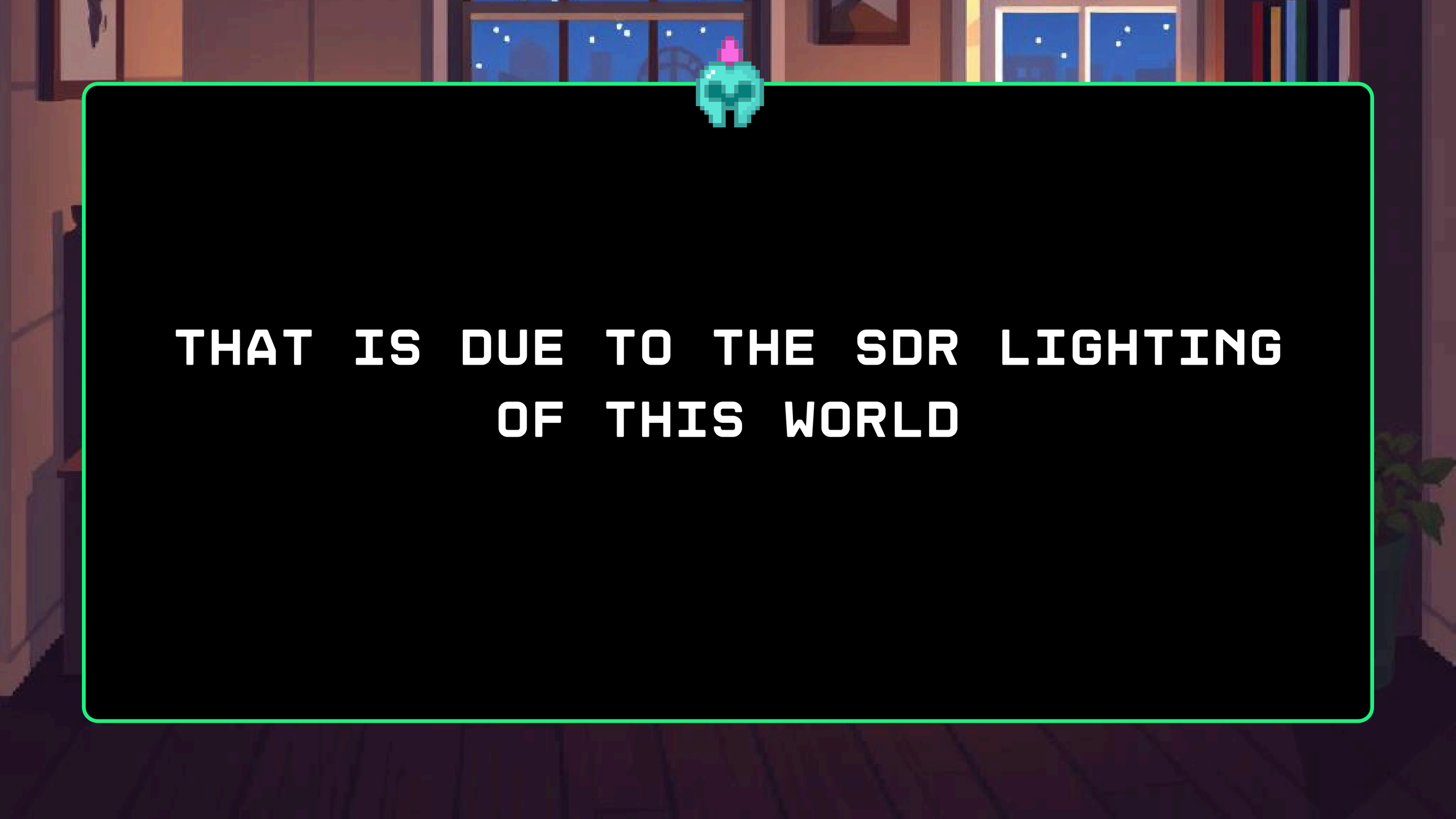
GAME OVER

GO TO LAST CHECKPOINT


YOU :
I don’t have time. Guess I’m taking the short path today.

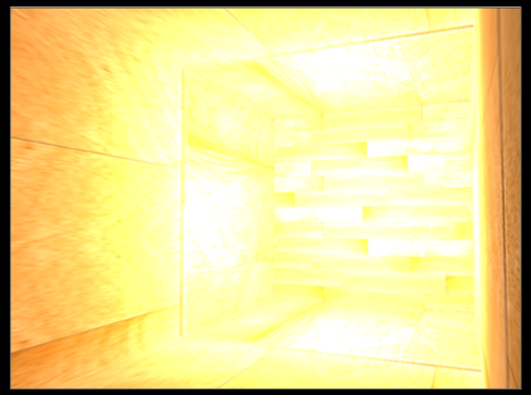

YOU MOVE FORWARD AND CRASH INTO
A POLE


YOU :
Man! why didn’t I see the pole?


THAT IS DUE TO THE SDR LIGHTING
OF THIS WORLD

HDR LIGHTING

And then god said, “Let there be better light”



When many pixel brightness intensities reach a value of more than 1, they are all clipped to 1. As a result, all those pixels appear at the same intensity, destroying the image quality. Resulting in an image like this.
One proposed solution is scaling down all pixels, which results in a sunglasses-like look

By allowing fragment colors to exceed 1.0 we have a much higher range of color values available to work in known as high dynamic range (HDR).

HOW IT WORKS

"Let's not oversaturate things"

- High dynamic range was originally only used for photography where a photographer takes multiple pictures of the same scene with varying exposure levels, capturing a large range of color values. Combining these forms a HDR image where a large range of details are visible based on the combined exposure levels, or a specific exposure it is viewed with.
- We allow for a much larger range of color values to render to, collecting a large range of dark and bright details of a scene, and at the end we transform all the HDR values back to the low dynamic range (LDR) of [0.0, 1.0].

We use tone-mapping algorithms like Reinhard tone mapping

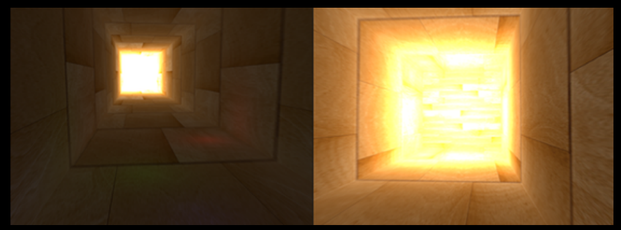
REINHARD TONE MAPPING

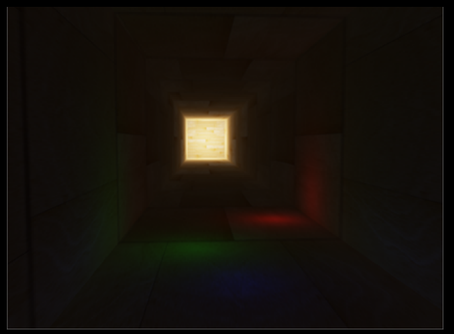
"Let's not oversaturate things"

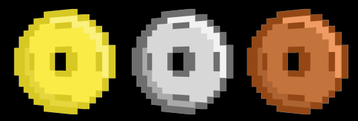
- Reinhard tone mapping involves dividing the entire HDR colour values into LDR colour values.
- The Reinhard tone mapping algorithm evenly balances out all brightness values onto LDR.

```
void main()
{
    const float gamma = 2.2;
    vec3 hdrColor = texture(hdrBuffer,
        TexCoords).rgb;
      // reinhard tone mapping
    vec3 mapped = hdrColor / (hdrColor +
                        vec3(1.0));
    FragColor = vec4(mapped, 1.0);
}
```

Result:

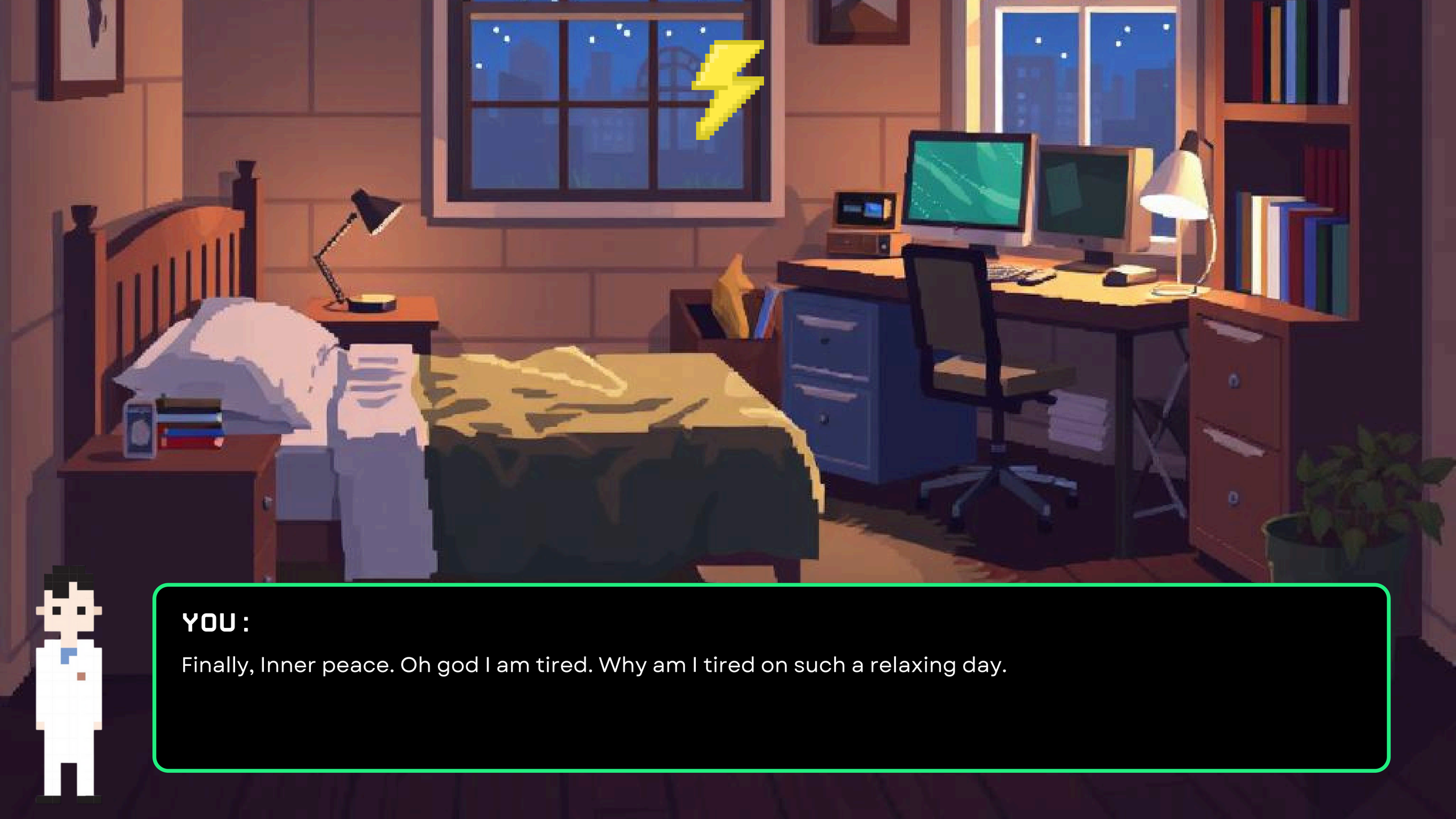


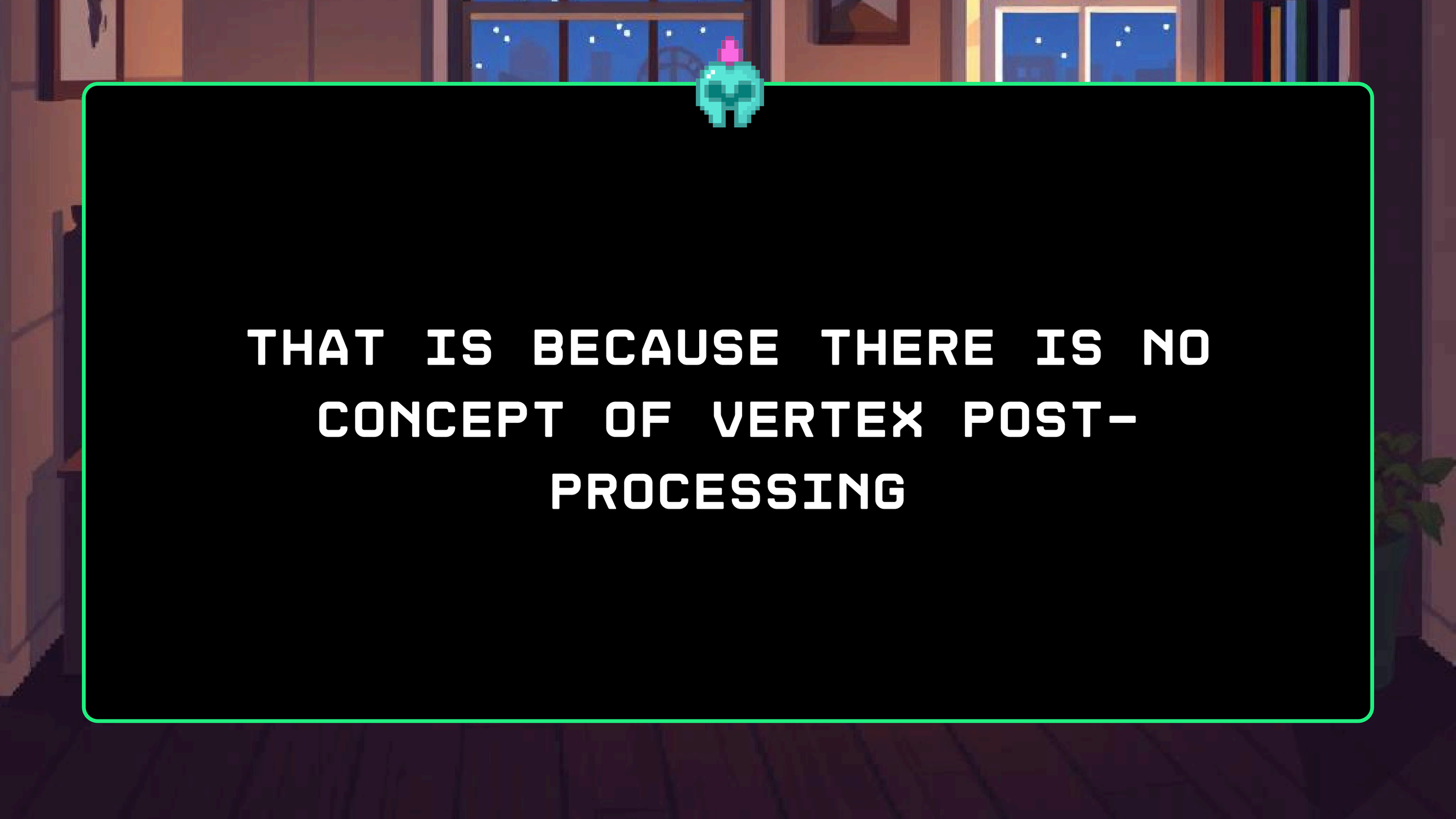




NOW YOU KNOW ABOUT HDR LIGHTING

This lighting method is widely used in the gaming and video industry, as videos with HDR are often much better perceived than non-HDR ones.
You enter your room and lie down.


YOU :
Finally, Inner peace. Oh god I am tired. Why am I tired on such a relaxing day.


THAT IS BECAUSE THERE IS NO CONCEPT OF VERTEX POST-PROCESSING

TRANSFORM FEEDBACK

Ben 10 Nostalgia

- Transform Feedback is an OpenGL feature that captures the output of the vertex shader (or geometry shader) and writes it directly into a buffer without passing through the fragment stage.
- This allows us to store transformed vertices for later use, without sending them to the screen

The process involves:

1. Enable transform feedback mode
2. Bind a buffer to store the output
3. Define which variables to capture
4. Render geometry (without rasterization)
5. Retrieve transformed data

```
glEnable(GL_RASTERIZER_DISCARD);

glBindBufferBase(GL_TRANSFORM_FEEDBACK_BUFFER,
        0, feedbackBuffer);
glBeginTransformFeedback(GL_POINTS);

glDrawArrays(GL_POINTS, 0, NUM_VERTICES);

glEndTransformFeedback();
glDisable(GL_RASTERIZER_DISCARD);
```

CLIPPING

That reminds me, Clippy just got clipped out of existence, didn't he?

- Primitives generated by previous stages are collected and then clipped to the view volume. Each vertex has a clip-space position (the gl_Position output of the last Vertex Processing stage).
- While the clipping is automatically enabled by OpenGL, we can add an additional manual clipping factor by using:

```
float* gl_ClipDistance;
// This array is a built-in variable
glEnable(GL_CLIP_DISTANCEi);
// i corresponds to ith element
// of the gl_ClipDistance array
// gl_ClipDistance[i] > 0 implies
// ith point is inside
// the clipping volume
```

PERESPECTIVE DIVISION

Its a cultural divide, Imma get it on the floor

- The clip-space positions returned from the clipping stage are transformed into normalized device coordinates (NDC) via this equation:

$$\begin{pmatrix} x_{ndc} \\ y_{ndc} \\ z_{ndc} \end{pmatrix} = \begin{pmatrix} \frac{x_c}{w_c} \\ \frac{y_c}{w_c} \\ \frac{z_c}{w_c} \end{pmatrix}$$

VIEWPORT TRANSFORM

- Converts NDC (-1 to 1 range) into window coordinates (pixel space).
- Ensures that objects are mapped correctly onto the frame buffer.

$$\begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} = \begin{pmatrix} \frac{width}{2} x_{ndc} + x + \frac{width}{2} \\ \frac{height}{2} y_{ndc} + y + \frac{height}{2} \\ \frac{farVal - nearVal}{2} z_{ndc} + \frac{farVal + nearVal}{2} \end{pmatrix}$$

BACK TO LEVELS MENU

04 24 04

CONGRATULATIONS! YOU HAVE CLEARED THE HALL ARC!!


THANK YOU!
ANSHUMAAN MISHRA
24CS10082